

Exhibit 4

```

1 <!--#include virtual="/common/properties/stdheader.jsp"-->
    <% cJMBPropertyPage = new
com.control.jsp.green.web.jsp.controls.MicroBlockString(loc.getBinder(), "M012", "I/O Points");%>
    <!--#include virtual="/common/properties/equipintro.jsp"-->
5
    <% cJMBPropertyPage = new
com.control.jsp.green.web.jsp.controls.MicroBlockString(loc.getBinder(), "points", "I/O Points");%>
    <!--#include virtual="/common/properties/microblocks/mb_text_expandablebegin.jsp"-->
    <DIV style="height:10px; line-height:10px;">&nbsp;  </DIV>

10 <% cJMBPropertyPage = new
com.control.jsp.green.web.jsp.controls.MicroBlockString(loc.getBinder(), "M012", "I/O Points");%>
    <!--#include virtual="/common/properties/microblocks/mb_text_tablebegin.jsp"-->

    <% cJMBPropertyPage = new
com.control.jsp.green.web.jsp.controls.MicroBlockString(loc.getBinder(), "CTRL", "Property");%>
    <!--#include virtual="/common/properties/microblocks/mb_bai.jsp"-->
15
    <% cJMBPropertyPage = new
com.control.jsp.green.web.jsp.controls.MicroBlockString(loc.getBinder(), "M006", "Property");%>
    <!--#include virtual="/common/properties/microblocks/mb_bao.jsp"-->

    <% cJMBPropertyPage = new
com.control.jsp.green.web.jsp.controls.MicroBlockString(loc.getBinder(), "M007", "Property");%>
    <!--#include virtual="/common/properties/microblocks/mb_bao.jsp"-->
20
    <% cJMBPropertyPage = new
com.control.jsp.green.web.jsp.controls.MicroBlockString(loc.getBinder(), "M012", "I/O Points");%>
    <!--#include virtual="/common/properties/microblocks/mb_text_tableend.jsp"-->

25 <% cJMBPropertyPage = new
com.control.jsp.green.web.jsp.controls.MicroBlockString(loc.getBinder(), "M012", "I/O Points");%>
    <!--#include virtual="/common/properties/microblocks/mb_text_expandableend.jsp"-->

    <BR>

30 <% cJMBPropertyPage = new
com.control.jsp.green.web.jsp.controls.MicroBlockString(loc.getBinder(), "control", "Control Loops");%>
    <!--#include virtual="/common/properties/microblocks/mb_text_expandablebegin.jsp"-->
    <DIV style="height:10px; line-height:10px;">&nbsp;  </DIV>

    <% cJMBPropertyPage = new
com.control.jsp.green.web.jsp.controls.MicroBlockString(loc.getBinder(), "M004", "Direct Acting Loop");%>
35 <!--#include virtual="/common/properties/microblocks/mb_pid2da.jsp"-->
    <BR>

    <% cJMBPropertyPage = new
com.control.jsp.green.web.jsp.controls.MicroBlockString(loc.getBinder(), "M005", "Reverse Acting Loop");%>
    <!--#include virtual="/common/properties/microblocks/mb_pid2ra.jsp"-->

```

```
40 <% cjmBPropertyPage = new
com.control.j.green.web.jsp.controls.MicroBlockString(loc.getBinder(), "M012", "I/O Points");%>
<!--#include virtual="/common/properties/microblocks/mb_text_expandableend.jsp"-->
<BR>

45 <% cjmBPropertyPage = new
com.control.j.green.web.jsp.controls.MicroBlockString(loc.getBinder(), "M012", "Setpoints");%>
<!--#include virtual="/common/properties/microblocks/mb_text_sep.jsp"-->

<% cjmBPropertyPage = new
com.control.j.green.web.jsp.controls.MicroBlockString(loc.getBinder(), "M002", "Property");%>
50 <!--#include virtual="/common/properties/microblocks/mb_setpt.jsp"-->
<BR>&nbsp;
<BR>
<% cjmBPropertyPage = new
com.control.j.green.web.jsp.controls.MicroBlockString(loc.getBinder(), "M014", "Temperature Colors");%>
55 <!--#include virtual="/common/properties/microblocks/mb_tstclr.jsp"-->
<!--#include virtual="/common/properties/stdfooter.jsp"-->
```

Exhibit 5

```

1  <HTML>
   <HEAD>
   <LINK ID="ctrlStyles" REL=STYLESHEET TYPE="text/css"
   HREF="/common/pagestyles/standard/controlstyles.css">
   <%@import="com.controlj.green.web.jsp.controls.*, com.controlj.green.core.data.CoreNode"%>
10  <% Location loc = new Location(request); %>
   <% String cJMicroBlock; %>
   <% String cJMBString; %>
   <% MicroBlockString cJMBPropertyPage; %>

15  <% boolean cJInTable = false; %>

   <SCRIPT LANGUAGE="JavaScript" src="/common/debug/cjdebugpub.js"></SCRIPT>

   <SCRIPT LANGUAGE="JavaScript" src="/common/util/CJElementHelper.js"></SCRIPT>
15  <SCRIPT LANGUAGE="JavaScript" src="/common/util/BrowserApi.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/exception.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/CJEvent.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/primitive.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlBase.js"></SCRIPT>
20  <SCRIPT LANGUAGE="JavaScript" src="/common/controls/stylemgr.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/TextValidation.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/widgetTextInput.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/iConHelper.js"></SCRIPT>
25  <SCRIPT LANGUAGE="JavaScript" src="/common/controls/widgetImageButton.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/widgetBandsBar.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlBase.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlImageButton.js"></SCRIPT>
30  <SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlTextInput.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/widgetDroplist.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlDroplist.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlDateInput.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlTimeInput.js"></SCRIPT>
35  <SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlSlider.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlBacnetOID.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlColorMap.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlSetPoint.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlImageMap.js"></SCRIPT>
40  <SCRIPT LANGUAGE="JavaScript" src="/common/controls/widgetImageAreaLabel.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlColorMap.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/controls/ControlMinSecInput.js"></SCRIPT>
   <SCRIPT LANGUAGE="JavaScript" src="/common/util/classHelper.js"></SCRIPT>

   <!--#include virtual="/common/util/stdActionPage.jsp" -->
45  </HEAD>

   <BODY onLoad="onPageLoaded()" onUnload="CJElementHelper.flushList()"

```

```
style="background-image:url(/common/pagestyles/standard/GreyText
50 <DIV ID="cJouter" style="padding:10px;">
    <IFRAME NAME="primitivesubmit" ID="primitivesubmit" WIDTH="0" HEIGHT="0"
SRC="/common/darkblank.html" style="visibility:hidden;"></I
<IFRAME NAME="primitiveupdate" ID="primitiveupdate" WIDTH="0" HEIGHT="0"
SRC="/common/darkblank.html" style="visibility:hidden;"></I
<IFRAME NAME="jumprequest" ID="jumprequest" WIDTH="0" HEIGHT="0" SRC="/common/darkblank.html"
style="visibility:hidden;"></IFRAME>
55
```

Exhibit 6

```
1 </DIV>
  <SCRIPT language="JavaScript">
    getupdate(true); // Get update and force field refresh
    top.setActBusy(false);
5 </SCRIPT>
  </BODY>
</HTML>
```


Exhibit 7

```

1 <SCRIPT LANGUAGE="JavaScript">
  /*****
    AUTOMATED LOGIC CORPORATION
    Copyright (c) 1999 - 2000 All Rights Reserved
    This document contains confidential/proprietary information.
    *****/
5  @(#)StdActionPage.jsp
   Initial Author: Steve Appling
10  Description: Contains various functions that commonly used on action
    pages.
   $Log: $
15  *****/

   var AdvPropertiesRule;

20  function onPageLoaded()
   {
     if (window.parent && window.parent.mainLoaded)
     {
       var cjOuter = document.getElementById("cjOuter");
       CjAssert(cjOuter, "cjOuter container not found");
       window.parent.actionFrameLoaded(window, cjOuter);
     }
     PrimitiveBase.addUIListener(this);
30   var ruleset = document.styleSheets["ctrlstyles"].rules;
     if (ruleset)
     {
       AdvPropertiesRule = ruleset[0].style;
     }
35 }

   var _cjPageRefresh = 4000;
   var _cjUpdateCount = 0;
   var _cjGetValuesFromField=false;
40  function updateFinished()
   {
     //window.status=_cjUpdateCount++;
     setTimeout("getUpdate(_cjGetValuesFromField)", _cjPageRefresh);
   }
45

   function getUpdate(refreshField)
   {
50     window.parent.setActBusy(true);

```

```

        document.getElementById("primitiveupdate").src="/common/servlet/primitiveupdate?wbs="+<%=
loc.getWBS().getId() %>+"&getFieldValues
    }
}
55
function valueChangedByUser(from)
{
    window.parent.currentServerPage = this;
    window.parent.setActionButtonState(false);
60 }
function postChanges()
{
    var changes = PrimitiveBase.changedPrimitives();
    var commandList = "/common/servlet/primitivesubmit?wbs="+<%= loc.getWBS().getId() %>;
    for (var i=0; i<changes.length; i++)
    {
        var newChange = changes[i].getId()+"="+escape(changes[i].toString());
        commandList += "&" + newChange;
    }
    //alert(commandList);
    window.parent.setActBusy(true);
    document.all.primitivesubmit.src = commandList;
70 }
75 }
function cancelChanges()
{
    PrimitiveBase.resetAllChangedPrimitives();
80 }
function jumpToTreeLocation(treeVarName, location, page)
{
    window.parent.setNavBusy(true);
    var jumpRequest = "/common/servlet/treejumprequest" +
        "?wbs=<%= loc.getWBS().getId() %>" +
        "&arehere=<%= loc %>" +
        "&treename="+ treeVarName +
        "&path="+ location +
        "(page!=null?"&page="+page:"");
90
    //alert(jumpRequest);
    var jumpframe = document.getElementById("jumprequest");
    if (jumpframe)
    {
        document.all.jumprequest.src = jumpRequest;
95
    }
    else
    {

```

```
100     alert("Error: Action page contains no jumprequest IFRAME");
      }
    }
    var disableControls = new Object();
105  </SCRIPT>
```

Exhibit 8

1 /*****
AUTOMATED LOGIC CORPORATION
Copyright (c) 1999 - 2000 All Rights Reserved
This document contains confidential/proprietary information.

5 *****
@(#)primitive.js
Initial Author: Doyle Davidson

10 Description: standard primitives

\$Log: /green/webroot/common/controls/Primitive.js \$

15 * 39 12/14/99 11:41a Ddavidson
* Fixed exception message in addUListener

* 38 12/13/99 3:41p Ddavidson
* added isPageLocal().
* fixed table setColumn to do notifyListeners();

20 * made notifyListeners pass the "from" as a second param

25 * 37 12/08/99 1:24p Ddavidson
* Modified PrimitiveEnum has handle any "key" value. Doesn't have to be
* just numbers, can also be any string.

* 36 12/08/99 10:10a Jomisore
* Fix for getting number of rows when there is no data.

30 * 35 12/06/99 11:25a Ddavidson
* added code to notify listeners when all primitives are reset WHEN THE
* PRIMITIVE DIDN'T CHANGE.

35 * 34 12/03/99 5:59p Jomisore
* Fixed the string format for tables.

* 33 11/24/99 1:45p Jomisore
* Fixed merge conflicts.

40 * 32 11/24/99 1:39p Jomisore
* Table primitive 1st version.

* 31 11/18/99 12:57p Tirish

* 30 11/17/99 8:39a Tirish

45 * 29 11/16/99 9:51a Ddavidson

```

* Made clearChanged() an instance method. This allows a primitive to
* override it.
* PrimitiveBase.clearChangedFlag() now invokes instance method.
50 *
* 28 11/09/99 8:49a Ddavidson
* fixed PrimitiveBitString_setBit to invoke underlying setValue() call in
* order to trigger listeners.
*
* 27 11/05/99 9:56a Sappling
55 * Added PrimTable basic support
*
* 26 11/04/99 10:14a Ddavidson
* added "changed=false" to resetValue() method.
60 *
* 25 11/03/99 9:42a Jomisore
* Changed it so that findPrimitive throws an exception again when the
* primitive is not found.
*
* 24 11/01/99 4:12p Ddavidson
65 * Fixed notifyUIListener to pass the primitive being changed in the
* callback
*
* 23 10/07/99 3:14p Sappling
70 * Removed exception when primitive not found
*
* 22 9/20/99 3:45p Wdover
* Added clearChangedFlag static method to PrimitiveBase
75 *
* 21 9/15/99 5:11p Wdover
* Modified to disable cancel on local primitives
*
* 20 9/15/99 5:04p Wdover
* Modified to disable accept/cancel on primitives not connected to core
80 *
* 19 9/14/99 10:39a Wdover
* Modified to use negative max value for null minimum number values
*
* 18 9/10/99 4:43p Jomisore
85 * Fixed a problem where when setting the "minimum" restriction it was
* checking against "this.maximum" which wasn't already set.
*
* 17 9/09/99 3:36p Wdover
* Modified to enable bad value checking
90 *
* 16 9/09/99 2:20p Jomisore
* Fixed a problem with checking precision restrictions of numbers.

```



```

140 * Javascript primitive objects
    *****/
    /*****
145 *****
    CLASS: PrimitiveBase
    ATTRIBUTES:
150         id      String      A unique string identifying the primitive instance

        listeners  Array      An array of object supporting the "valueChanged"
                                function that are notified when the value changes.
        uiListeners Array      An array of objects supporting the "valueChangedByUser"
                                function that are notified when the value changes
                                as a result of user actions.
        data        object      The current valid value of the primitive
        origData    object      The original/last value specified from CORE
        changed     boolean     True if the value has been set using setValue
                                note that this will disable setMasterValue
160 *****
    *****/

function PrimitiveBase( id, writable, initData )
165 {
    if ( id == null ) // handle case of assignment to prototype
        return;

    this.id = id;
    this.writable = writable;
    this.listeners = new Array;
    // if the initData is a string or a string object
    // convert it to it's internal representation
    if( initData!=null && typeof initData.valueOf() == "string")
175 {
        initData=this.fromString(initData);
    }
    this.origData = initData;
    this.resetData();
180
    if ( PrimitiveBase.allPrimitives[id] != null )
    {
        throw new CJException.DuplicateItem(

```

```

185         "Primitive named \" + id + "\" already exists" );
    }
    PrimitiveBase.allPrimitives[id] = this;
    this.changed = false;
    this.badValue = false;
190     this.pageLocal = false;
}

/*****
195     Member accessors:
*****/

// return primitive's ID
function _PrimitiveBase_getId()
{
    return this.id;
200 }

// return true if the primitive is different from it's original value
function _PrimitiveBase_isChanged()
205 {
    return (this.changed);
}

// return true if primitive is undefined (value is null)
function _PrimitiveBase_isUndefined()
210 {
    return ( this.data == null );
}

215 // return true if primitive is pageLocal
function _PrimitiveBase_isPageLocal()
{
    return ( this.pageLocal );
}
220 // return true if one can write to the primitive
function _PrimitiveBase_isWritable()
{
    return this.writable;
225 }

// return true if value from primitive is bad
function _PrimitiveBase_hasBadValue()
{
230     return this.badValue;
}

```

```

    }

    // return the primitive's value
    function _PrimitiveBase_getValue()
235 {
    return this.data;
}

    // clear the "changed" flag
    function _PrimitiveBase_clearChanged( )
240 {
    {
        this.changed = false;
    }
}

245 // return the primitive's value as a string
    // this is the default method
    function _PrimitiveBase_toString()
    {
        //There is a little bit of a debate over what this should return ""
        //when the value is null. For now we are returning the empty string ""
        if ( this.data == null )
            return "";
        else
            return this.data.toString();
255 }

    // converts the string given into the proper format
    //This default method does nothing
    function _PrimitiveBase_fromString(text)
    {
260     return text;
    }

    // checks if the value is valid
    //This default method will accept any value
    //and is here only as a placeholder
    function _PrimitiveBase_validateValue( test )
265 {
    }

270 /*****
    Class : PrimitiveBase
    Method: notifyListeners
    Params: Listener except
    Return: none
    275 Description: walks the list of listeners and invokes the
        "valueChanged() method for each one. If an 'except' listener is

```

```

specified, then that listener, if present, will be skipped and
therefore not notified.
*****/
280 function _PrimitiveBase_notifyListeners( except )
    {
        for ( var i=0; i<this.listeners.length; i++ )
        {
            if ( this.listeners[i] != null && except != this.listeners[i] )
            {
                this.listeners[i].valueChanged( this, except );
            }
        }
    }
285
290
295
299 *****/
    Class : PrimitiveBase
    Method: notifyUIListeners
    Params: Primitive prim      The primitive being changed.
           Object from         The object causing the change
    Return: none
    Description: walks the list of uilisteners and invokes the
                 "valueChangedByUser() method for each one.
*****/
300 function _PrimitiveBase_notifyUIListeners( prim, from )
    {
        for ( var i=0; i < PrimitiveBase.uilisteners.length; i++ )
        {
            if ( PrimitiveBase.uilisteners[i] != null )
            {
                PrimitiveBase.uilisteners[i].valueChangedByUser( prim, from );
            }
        }
    }
305
310
315
319 *****/
    Class : PrimitiveBase
    Method: setMasterValue
    Params: object newValue
    Return: none
    Description: Sets the master value of the primitive unconditionally
                 and the current value only if never set by the user.
                 It then notifies all listeners of the change.
320
325 *****/
    Essentially don't set the current value the first time in if the
    user has already set a current value. Meaning he has changed the
    value before the master value was initially loaded.

```

```

325 *****_PrimitiveBase_setMasterValue( newData )
    {
        // If master has been set before, or current was never set and
        // the user has not changed the value, then set the current value.
        newData = this.fromString(newData);
330         this.origData = newData;

        if (( this.origData != null || this.data == null ) && !this.changed)
        {
335             this.resetData();
            this.changed = false;
            this.notifyListeners();
        }
340     }
    /*****
    Class : PrimitiveBase
    Method: setMasterUndefined
    Params:
    Return: none
    Description: Sets the master value of the primitive to the undefined state
    and the current value only if never set by the user.
    It then notifies all listeners of the change.
345 *****_PrimitiveBase_setMasterUndefined()
    {
350         this.origData = null;

355         if (!this.changed)
        {
            this.resetData();
            this.changed = false;
            this.notifyListeners();
360         }
    }

    /*****
    Class : PrimitiveBase
    Method: setBadValue
    Params: boolean badValueFlag - new value of badValueFlag
    Return: none
365

```

370 Description: Sets the bad value flag of the primitive unconditionally if different from its current value and if the user has not changed the primitive. It then notifies all listeners of the change so that they can act appropriately.

```
375 *****/
function _PrimitiveBase_setBadValue( badValueFlag )
{
    if ((this.badValue != badValueFlag) && !this.changed)
    {
        this.badValue = badValueFlag;
        this.notifyListeners();
    }
}
```

```
385 // Set the primitive's value - returns 'true' if successful.
*****/
Class : PrimitiveBase
Method: setValue
Params: object newData, Listener from
Return: none
390 Description: Attempts to set the current value of the primitive.
The primitive must be writable and the value must be valid.
If either of these conditions fails, an exception will be thrown.
Otherwise, the value is set and the listeners are notified. The
listener specified in the 'from' value, if any, will be excepted
from the notification.
395 Exceptions: ReadOnly
*****/
function _PrimitiveBase_setValue( newData, from )
{
```

```
400 // if writable, set new value and call listeners
if ( this.writable == false )
{
```

```
405     throw new CJException.ReadOnly(
        "Cannot set value for primitive \"" + this.id + "\"");
    }
    this.validateValue( newData );
```

```
    // update value
    this.data = newData;
    this.changed = (this.data != this.origData);
    this.notifyListeners( from );
    if (!this.pageLocal)
    {
        PrimitiveBase.notifyUILListeners(this, from);
    }
}
```

```

415     }
    }

    // Set the primitive's value - returns 'true' if actually changed.
    //*****
420     Class : PrimitiveBase
    Method: resetValue
    Params: Listener from
    Return: Boolean - true if the value changed
    Description: Resets the current value to the master value if it is
425     different. If a change occurs, the listeners will be notified with
    the possible exception of the listener that reset the value.
    *****
    function _PrimitiveBase_resetValue(from)
    {
430         // if writable, set new value and call listeners
        if (this.writable && this.changed == true)
        {
            // update value
            this.resetData();
            this.changed = false;
            this.notifyListeners( from );
            if (!this.pageLocal)
            {
440                 PrimitiveBase.notifyAllListeners(this, from);
            }
            return true;
        }
        else
        {
            return false;
445     }
    }

    // add listener to the primitive. It must have "valueChanged" function
    //*****
450     Class : PrimitiveBase
    Method: addListener
    Params: Listener who
    Return: none
    Description: Adds the specified to the list of listeners if it isn't
455     already on the list. A listener is defined as any object that has
    a "valueChanged" function.
    *****
    function _PrimitiveBase_addListener( who )
    {
460         CAssert( who != null, "listener is null" );
    }

```

```

465 // don't add it object doesn't have a valueChanged method
    if ( who.valueChanged == null )
    {
        throw new CJException.InvalidClass(
            "PrimitiveBase_addListener_" +
            "attempt to add listen with no valueChanged() method" );
    }

470 // check for already added
    var i = 0;
    for ( i=0; i<this.listeners.length; i++ )
    {
        if ( this.listeners[i] == who )
            return;
    }

475 // add to end
    this.listeners[this.listeners.length] = who;
}

480 /*****
    Class : PrimitiveBase
    Method: removeListener
    Params: Listener who
    Return: none
    Description: Removes the specified listens from the list of listeners.
    *****/
function _PrimitiveBase_removeListener( who )
{
    var i = 0;
    for ( i=0; i<this.listeners.length; i++ )
    {
        if ( this.listeners[i] == who )
            this.listeners[i] = null;
    }

495 }

500 // add uilistener to the primitive. It must have "valueChangedByUser" function
    /*****/
    Class : PrimitiveBase
    Method: addUIListener
    Params: Listener who
    Return: none
    Description: Adds the specified to the list of listeners if it isn't
    already on the list. A listened is defined as any object that has
    a "valueChanged" function.

505

```



```

*****/
function _PrimitiveBase_addUListener( who )
{
    510    CJAssert( who != null, "listened is null" );
    // don't add it object doesn't have a valueChanged method
    if ( who.valueChangedByUser == null )
    {
        throw new CJException.InvalidClass(
            515    "PrimitiveBase_addUListener - " +
            "attempt to add listen with no valueChangedByUser() method" );
    }

    // check for already added
    520    var i = 0;
    for ( i=0; i<PrimitiveBase.uListeners.length; i++ )
    {
        if ( PrimitiveBase.uListeners[i] == who )
        525    {
            return;
        }

        // add to end
        PrimitiveBase.uListeners[PrimitiveBase.uListeners.length] = who;
    }
    530    /*****
    Class : PrimitiveBase
    Method: removeUListener
    Params: Listener who
    Return: none
    Description: Removes the specified listens from the list of listeners.
    *****/
    function _PrimitiveBase_removeUListener( who )
    {
        540    var i = 0;
        for ( i=0; i<PrimitiveBase.uListeners.length; i++ )
        {
            if ( PrimitiveBase.uListeners[i] == who )
                PrimitiveBase.uListeners[i] = null;
        }
        545    }

    /*****
    Class : PrimitiveBase
    Method: findPrimitive (static class method)
    Params: String id
    Return: Primitive
    550

```

```

Description: Returns the primitive with the specified ID.
Exceptions: NullValue, NotFound
*****
555 function _PrimitiveBase_findPrimitive( id )
    {
        if ( id == null )
        {
            throw new CJException.NullValue(
                "PrimitiveBase_findPrimitive - id is NULL" );
        }

        if ( PrimitiveBase.allPrimitives[id] == null )
        {
            throw new CJException.NotFound(
                "PrimitiveBase_findPrimitive - primitive for id \"\"
                + id + "\" not found" );
        }

        return PrimitiveBase.allPrimitives[id];
    }
}

/*****
575 Class : PrimitiveBase
Method: changedPrimitives: (static class method)
Params: none
Return: Array of primitives
Description: Rreturns an array of all the primitives that have been
changed. This is determined by the isChanged() method.
*****
580 function _PrimitiveBase_changedPrimitives( )
    {
        var changed = new Array;
        for ( var id in PrimitiveBase.allPrimitives )
        {
            var prim = PrimitiveBase.allPrimitives[id];
            if ( prim.isChanged() )
                changed[changed.length] = prim;
        }

        return changed
    }
}

595 /****
Class : PrimitiveBase
Method: setPrimitiveValue (static class method)
Params: String id, object newData
*****

```

```

Return: none
Description: Finds the specified primitive by ID and then invokes the
  setValue() method.
Exceptions: NullValue, NotFound
*****function _PrimitiveBase_setPrimitiveValue( id, newData, from )
605 {
    // findPrimitive will throw exception if ID is bad
    PrimitiveBase.findPrimitive( id ).setValue( newData, from );
}

610 /*****
Class : PrimitiveBase
Method: setPrimitiveBadValue (static class method)
Params: String id, boolean badValueFlag
Return: none
Description: Finds the specified primitive by ID and then invokes the
  setBadValue() method.
Exceptions: NullValue, NotFound
*****function _PrimitiveBase_setPrimitiveBadValue( id, badValueFlag )
620 {
    // findPrimitive will throw exception if ID is bad
    PrimitiveBase.findPrimitive( id ).setBadValue( newData, from );
}

625 /*****
Class : PrimitiveBase
Method: resetAllChangedPrimitives (static class method)
Params: from - object that called
Return: none
Description: Resets all changed primitives to the original values
Exceptions:
*****function _PrimitiveBase_resetAllChangedPrimitives(from)
635 {
    for ( var id in PrimitiveBase.allPrimitives )
    {
        var prim = PrimitiveBase.allPrimitives[id];
        if ( prim.isChanged() && (!prim.pageLocal))
            prim.resetValue(from);
        else
        {
            // And notify all listeners even if nothing changes.
            // This should reset any invalid controls too.
            prim.notifyListeners( null );
640

```

```

645     }
        }
    }

    /*****
650     Class : PrimitiveBase
        Method: setPrimitiveMasterValue (static class method)
        Params: String id, object newData
        Return: none
        Description: Finds the specified primitive by ID and then invokes the
655         setMasterValue() method.
        Exceptions: NullValue, NotFound

        NOTE: THIS METHOD SHOULD ONLY BE INVOKED INDIRECTLY FROM THE CORE VIA
        A SERVLET/JSP PAGE!!!!
660     *****/
    function _PrimitiveBase_setPrimitiveMasterValue( id, newData )
    {
        // findPrimitive will throw exception if ID is bad
        PrimitiveBase.findPrimitive( id ).setMasterValue( newData );
665    }

    /*****
        Class : PrimitiveBase
        Method: ClearChangedFlag (static class method)
670     Params: String id
        Return: none
        Description: Finds the specified primitive by ID and then clears its
        changed flag. This signifies that core has received the update from
        the user.
        Exceptions: NullValue, NotFound
675

        NOTE: THIS METHOD SHOULD ONLY BE INVOKED INDIRECTLY FROM THE CORE VIA
        A SERVLET/JSP PAGE!!!!
        *****/
    function _PrimitiveBase_clearChangedFlag( id )
    {
        // findPrimitive will throw exception if ID is bad
        PrimitiveBase.findPrimitive( id ).clearChanged();
680    }

    /*****
        Class : PrimitiveBase
        Method: helper_to8HexString
        Params: none
685     Return: 8 char hex string.
    *****/

```

```

Description: Converts the current data value to a 8 char (4 byte)
hex string with leading zeros.
*****/
function _PrimitiveBase_helper_to8HexString( )
695 {   if ( this.data == null )
        {
            return "00000000";
        }

700     var rval = "00000000" + this.data.toString(16);
        return rval.substr( rval.length-8, 8 );
    }

705 /*****
    Class : PrimitiveBase
    Method: setPageLocal
    Params: state
    Return: none
    Description: accessor to set the pageLocal state. When set to true,
                 the primitive will not call uilisteners when it changes value.
    *****/
    function _PrimitiveBase_setPageLocal(state)
    {
715         this.pageLocal = state;
    }

    /*****
    Class : PrimitiveBase
    Method: resetData
    Params:
    Return: none
    Description: Sets this.data to the master value.
    *****/
    function _PrimitiveBase_resetData()
    {
730         this.data = this.origData;
    }

    PrimitiveBase.prototype.PrimitiveBase = PrimitiveBase;
    PrimitiveBase.prototype.toString = _PrimitiveBase_toString;
    PrimitiveBase.prototype.fromString = _PrimitiveBase_fromString;

```

```

PrimitiveBase.prototype.getId
PrimitiveBase.prototype.isChanged
PrimitiveBase.prototype.isWritable
740 PrimitiveBase.prototype.isUndefined
PrimitiveBase.prototype.isPageLocal
PrimitiveBase.prototype.clearChanged
PrimitiveBase.prototype.validateValue
PrimitiveBase.prototype.setValue
745 PrimitiveBase.prototype.setMasterValue
PrimitiveBase.prototype.setMasterUndefined
PrimitiveBase.prototype.hasBadValue
PrimitiveBase.prototype.setBadValue
750 PrimitiveBase.prototype.resetValue
PrimitiveBase.prototype.addListener
PrimitiveBase.prototype.removeListener
PrimitiveBase.prototype.notifyListeners
PrimitiveBase.prototype.setPageLocal
755 PrimitiveBase.prototype.resetData

PrimitiveBase.findPrimitive
PrimitiveBase.changedPrimitives
PrimitiveBase.setPrimitiveValue
760 PrimitiveBase.setPrimitiveBadValue
PrimitiveBase.setPrimitiveMasterValue
PrimitiveBase.setPrimitiveMasters
PrimitiveBase.addUIListener
PrimitiveBase.removeUIListener
765 PrimitiveBase.resetAllChangedPrimitives
PrimitiveBase.clearChangedFlag

PrimitiveBase.allPrimitives = new Array;
PrimitiveBase.listeners = new Array;
770 /*****
****
**** END Primitive CLASS
****
****
****
775

780

```

780

```

*****
785  CLASS: PrimitivesString
    The string class is a simple primitive whos data type is a string

    Restrictions:
        maxlen - max string length (null = no limit)
*****
function PrimitivesString( id, writable, initData, maxlen )
795 {
    this.setMaxLength(maxlen);
    this.PrimitiveBase( id, writable, initData );
}

800 /*****
    Class : PrimitivesString
    Method: getMaxLength, hasMaxLength, setMaxLength and validateValue
    Params: varies
    Return: varies
    805 Description: The restriction for the PrimitivesString is its length
                    these methods handle this restriction.
*****
function _PrimitivesString_getMaxLength()
{
    810     if ( this.maxlength == null )
        return Number.MAX_VALUE;
    else
        return this.maxlength;
}

815 function _PrimitivesString_hasMaxLength()
{
    return this.maxlength != null;
}

820 function _PrimitivesString_setMaxLength(maxlen)
{
    if ( maxlen < 0 )
    {
        825     throw new CJException.InvalidValue(
        "max length " + maxlen + " must be >= 0" );
    }
}

```

```

830     this.maxLength=maxlen;
    }

    /*****
    Class : PrimitiveString
    Method: validateValue
    Params: String test
    Return: null
    Description: checks the test value against the primitive's restrictions
    *****/
    function _PrimitiveString_validateValue( test )
    {
        //check the test value's length.
        if ( this.hasMaxLength() )
        {
            if( test.length > this.getMaxLength() )
                throw new CJException("string of length " + test.length +
                                     " exceeds limit of " + this.getMaxLength());
        }
    }

850 PrimitiveString.prototype = new PrimitiveBase();
    PrimitiveString.prototype.validateValue= _PrimitiveString_validateValue;
    PrimitiveString.prototype.maxLength = null;
    PrimitiveString.prototype.getMaxLength = _PrimitiveString_getMaxLength;
    PrimitiveString.prototype.hasMaxLength = _PrimitiveString_hasMaxLength;
855 PrimitiveString.prototype.setMaxLength = _PrimitiveString_setMaxLength;

860

    /*****
    *****/

865     CLASS: PrimitiveBoolean

    The boolean class is a simple primitive whoses data type is a Boolean

    Restrictions:
    none

870

    *****/
    *****/
    function PrimitiveBoolean( id, writable, initdata )

```



```

875 {      this.PrimitiveBase( id, writable, initData );
      }

/*****
880      Class : PrimitiveBoolean
      Method: fromString
      Params: String text
      Return: Boolean
      Description: converts the string given into a boolean
885 *****/
      function _PrimitiveBoolean_fromString(text)
      {
          switch(text)
          {
              //returns a null if given a null
              case null: return null;
              //returns proper value when correct string is passed
              case "true": return true;
              case "false": return false;
              //throws an exception if an invalid string is passed
              default: throw new CJException.InvalidValue("\ntrue\" or \"false\". It received \""+text+"\"");
          }
      }

/*****
900      Class : PrimitiveBoolean
      Method: validateValue
      Params: Boolean test
      Return: null
      Description: checks the test value against the primitive's restrictions
905 *****/
      function _PrimitiveBoolean_validateValue( test )
      {
          //check if the test value is not one of the valid values.
          if ( test!=null && test!=true && test!="true" && test!="false" && test!=false )
          {
              throw new CJException.InvalidValue("Boolean must be true, false or null recieved \""+ test + "\".");
          }
      }

915 }
PrimitiveBoolean.prototype = new PrimitiveBase();
PrimitiveBoolean.prototype.fromString = _PrimitiveBoolean_fromString;
PrimitiveBoolean.prototype.validateValue = _PrimitiveBoolean_validateValue;

```



```

965 function _PrimitiveNumber_fromString(text)
{
    if(text==null){ return null;}
    //parse out the first number in the string.
    //must begin with a number.
    var number=parseFloat(text);
    //if a number was found return it
    if(!isNaN(number)){ return number;}
    //throw an exception if the conditions above were not met
    throw new CJException.InvalidValue("\nprimitiveNumber: fromString() must be given a string "+
    "\nbeginning with a number.It received \""+text+"\".");
}

970

975

/*****
Class : PrimitiveNumber
Method: getMaximum, hasMaximum and setMaximum
Params: varies
Return: varies
Description: One restriction for the PrimitiveNumber is its maximum
allowed value these methods handle this restriction.
*****/
980
985 function _PrimitiveNumber_getMaximum()
{
    if ( this.maximum == null )
        return Number.MAX_VALUE;
    else
        return this.maximum;
}

990

function _PrimitiveNumber_hasMaximum()
{
    return this.maximum != null;
}

995

function _PrimitiveNumber_setMaximum(max)
{
    if ( max < this.minimum )
    {
        throw new CJException.InvalidValue(
            "maximum value " + max + " must be >= the minimum value of "+this.minimum+"." );
    }
    this.maximum=max;
}

1000

1005

/*****
Class : PrimitiveNumber
Method: getMinimum, hasMinimum and setMinimum
*****/
1010

```

```

Params: varies
Return: varies
Description: One restriction for the PrimitiveNumber is its maximum
            allowed value these methods handle this restriction.
*****
1015 function _PrimitiveNumber_getMinimum()
    {
        if ( this.minimum == null )
            return -Number.MAX_VALUE;
        else
            return this.minimum;
    }

function _PrimitiveNumber_hasMinimum()
1025 {
    return this.minimum != null;
}

function _PrimitiveNumber_setMinimum(min)
1030 {
    if ( min > this.maximum )
    {
        throw new CJException.InvalidValue(
            "minimum value " + min + " must be <= the maximum value of "+this.maximum+"." );
1035     }
    this.minimum=min;
}

1040 /*****
Class : PrimitiveNumber
Method: getResNumerator, hasResNumerator and setResNumerator
Params: varies
Return: varies
Description: One restriction for the PrimitiveNumber is the resolution
            of it's numerator these methods handle this restriction.
*****
function _PrimitiveNumber_getResNumerator()
    {
        if(this.resnumerator==null) return 1.0;
        else return this.resnumerator;
    }

function _PrimitiveNumber_hasResNumerator()
1055 {
    return this.resnumerator != null;
}

```

```

    }
function _PrimitiveNumber_setResNumerator(resnum )
1060 {
    if ( resnum < 0 )
    {
        throw new CJException.InvalidValue("ResNumerator " + resnum + " must be >= 0" );
    }
    this.resnumerator=resnum;
1065 }
/*****
Class : PrimitiveNumber
Method: getResDenominator, hasResDenominator and setResDenominator
Params: varies
Return: varies
Description: One restriction for the PrimitiveNumber is the resolution
of it's denominator these methods handle this restriction.
*****/
1075 function _PrimitiveNumber_getResDenominator()
{
    if(this.resdenominator==null)
        return 1.0;
    else
        return this.resdenominator;
1080 }

function _PrimitiveNumber_hasResDenominator()
1085 {
    return this.resdenominator != null;
}

function _PrimitiveNumber_setResDenominator(resdenom )
1090 {
    if ( resdenom < 0 )
    {
        throw new CJException.InvalidValue("ResDenominator " + resdenom + " must be >= 0" );
    }
    this.resdenominator=resdenom;
1095 }

/*****
Class : PrimitiveNumber
Method: getTolerance, hasTolerance and setTolerance
Params: varies
*****/
1100

```

```

Return: varies
Description: One restriction for the PrimitiveNumber is the resolution
of it's numerator these methods handle this restriction.
*****
1105 function _PrimitiveNumber_getTolerance()
{
    if(this.tolerance==null)
        return 0.0;
    else
        return this.tolerance;
}

1115 function _PrimitiveNumber_hasTolerance()
{
    return this.tolerance != null;
}

1120 function _PrimitiveNumber_setTolerance(tol )
{
    if ( tol < 0 )
    {
        throw new CJException.InvalidValue("Tolerance " + tol + " must be >= 0" );
    }
    this.tolerance=tol;
}

/*****
1130 Class : PrimitiveNumber
Method: getPrecision, hasPrecision and setPrecision
Params: varies
Return: varies
Description: One restriction for the PrimitiveNumber is it's precision
these methods handle this restriction.
*****
1135 function _PrimitiveNumber_getPrecision()
{
    if(this.precision==null)
        return Number.MAX_VALUE
    else
        return this.precision;
}

1145 function _PrimitiveNumber_hasPrecision()
{
    return this.precision != null;
}

```

```

1150 function _PrimitiveNumber_setPrecision(prec )
{
    if ( prec < 0 )
    {
        throw new CJException.InvalidValue("Precision " + prec + " must be >= 0" );
    }
    this.precision=prec;
}

1160 /*****
Class : PrimitiveNumber
Method: validateValue
Params: Number test
Return: null
Description: checks the test value against the primitive's restrictions
*****/
function _PrimitiveNumber_validateValue( test )
{
    errorString="";
    // Convert to a number.
    test = new Number(test);

    if ( isNaN(test) == true )
    {
        errorString += "\"\" + test + "\" is not a number.\"";
        throw new CJException.InvalidValue(errorString);
    }

    //check if the number is between the min and max allowable values
    if ( this.hasMaximum() )
    {
        if( test > this.getMaximum() )
            errorString+="The Number "+ test +" exceeds limit of "+ this.getMaximum() +".\n";
    }
    if ( this.hasMinimum() )
    {
        if( test < this.getMinimum() )
            errorString+="The Number "+ test +" is below limit of "+ this.getMinimum() +".\n";
    }
    //if only the resnumerator is set treat as an integer resolution
    if ( this.hasResNumerator() && !this.hasResdenominator() )
    {
        res = this.getResNumerator();
    }
}

```

```

1195 min = ( this.hasMinimum() ? this.getMinimum() : 0 );
      baseVal = test - min;
      if ( baseVal % res != 0 )
      {
1200         errorString+="value \"" + test + "\" has an invalid resolution (" +
            res + ")\n";
      }
    }
  }
  // if both the resnumerator and the res denominator are set
  //treat as a fractional resolution
  if ( this.hasResNumerator() && this.hasResDenominator() )
  {
1210     min = ( this.hasMinimum() ? this.getMinimum() : 0.0 );
        num = this.getResNumerator();
        den = this.getResDenominator();
        testVal = test - min;

1215     // Determine nearest integral number of units of precision
        // (ex: if res=1/3, then 2.666 is 8 "thirds", so testVal=8
        testVal = (testVal * den) / num;

1220     // Now determine what the proper value should be
        // (ex: for testVal=8, we end up with 8/3, or 2.6666...
        // and determine the delta from that to compare against tolerance.
        // Note: You can't just take the value above since "0.34"
        // would result in 1.02, which would appear to be invalid
        // for a tolerance of 0.01. So recompute what the exact value
        // should be, and compare our tolerance to that.
        testVal = min + ( Math.round(testVal) * num / den );

1225     if ( Math.abs( testVal - test ) > this.getTolerance() )
    {
        errorString+="value \"" + test + "\" has an invalid resolution (" +
            num + "/" + den + "+-" + this.getTolerance() + ")\n";
    }
  }
  // now check number of significant digits
  if ( this.hasPrecision() )
  {
1235     exp = Math.pow( 10.0, this.getPrecision() );
        testVal = test * exp;
        if ( (test - Math.floor(testVal)/exp) > 0.0 )
        {
1240

```



```

errorString+="value \"\" + test + "\" may only have \" + this.getPrecision()
+ \" significant digits.\n";
    }
}
1245 if(errorString!="")
    {
        throw new CJException.InvalidValue(errorString);
    }
1250 }

PrimitiveNumber.prototype = new PrimitiveBase();
PrimitiveNumber.prototype.fromString = _PrimitiveNumber_fromString;
PrimitiveNumber.prototype.validateValue = _PrimitiveNumber_validateValue;
1255 PrimitiveNumber.prototype.maximum = null;
PrimitiveNumber.prototype.getMaximum = _PrimitiveNumber_getMaximum;
PrimitiveNumber.prototype.hasMaximum = _PrimitiveNumber_hasMaximum;
PrimitiveNumber.prototype.setMaximum = _PrimitiveNumber_setMaximum;
PrimitiveNumber.prototype.minimum = null;
1260 PrimitiveNumber.prototype.getMinimum = _PrimitiveNumber_getMinimum;
PrimitiveNumber.prototype.hasMinimum = _PrimitiveNumber_hasMinimum;
PrimitiveNumber. CLASS: PrimitiveSetElement

    An enumeration/bitstring has an associated "PrimitiveSetDefinition"
    that contains a list of "PrimitiveSetElement" objects. A set
    element is simply a key (typically a number) associated with a value.
    The key ('num') must be a string or a number.
    The value may be any Javascript object and care must be taken to
    ensure the user of the value is aware of the value's type.

1270 *****
*****/

// and number and associated value
1275 function PrimitiveSetElement( num, val )
{
    this.num = num;
    this.val = val;
}
1280

1285 /*****

```

```

1290 CLASS: PrimitiveSetDefinition
    A PrimitiveSetDefinition is a container of an array of
    PrimitiveSetElements. The elements are an ordered list.
    Restrictions:
        none
1295 ****/
function PrimitiveSetDefinition( id, writable, elements )
1300 {
1305     this.PrimitiveBase( id, writable, elements );
1310 }
1315 ****/
Class : PrimitiveSetDefinition
Method: getElementText
Params: val
Return: String
Description: Returns the string representation of the given element
            value. This value must match once of the enumeration values.
            Exceptions: NotFound
1320 ****/
function _PrimitiveSetDefinition_getElementText( val )
1325 {
1330     if ( this.data != null )
1335     {
1340         for ( var i=0; i<this.data.length; i++ )
1345         {
1350             if ( val == this.data[i].num )
1355             {
1360                 return this.data[i].val.toString();
1365             }
1370         }
1375         throw new CJException.NotFound(
1380             "PrimitiveSetDefinition_getElementText - element for value \"
            + val + "\" in set \"\" + this.id + "\" not found" );
1385     }
1390 }
1395 ****/
Class : PrimitiveSetDefinition
Method: getElementIndex
Params: val
Return: Number
1400 ****/

```



```

1405 this.setAllowed(allowed);
    // findPrimitive will throw exception if ID is bad
    this.enumDef = PrimitiveBase.findPrimitive( setId );
    this.enumDef.addListener( this );

    this.PrimitiveBase( id, writable, initIndex );
}
1410 /*****
    Class : PrimitiveEnum
    Method: valueChanged
    Params: prim
    Return: none
    Description: A PrimitiveEnum is a listener of it PrimitiveSetDefinition.
    Should the definition set change, then the representation of the
    value should also change. This does nothing more than propagate a
    change to the enum's listeners.
    *****/
1420 function _PrimitiveEnum_valueChanged( prim )
    {
        // propagate change
        this.notifyListeners( this );
    }
1425 }

// return set definition
1430 function _PrimitiveEnum_getSetDefinition( )
    {
        return this.enumDef;
    }

1435 // returned element's enum text
    function _PrimitiveEnum_getElementText( )
    {
        return this.enumDef.getElementText( this.data );
    }

1440 // return element's enum index
    function _PrimitiveEnum_getElementIndex( )
    {
        return this.enumDef.getElementIndex( this.data );
    }
1445 }
/*****

```

```

1450      Class : PrimitiveEnum
      Method: increment
      Params: Listener from
      Return: Boolean - if changed
      Description: Increments the enumeration value to the next value in
      the set definition. Returns false if already at the last element.
      *****/
1455  function _PrimitiveEnum_increment( from )
  {
      if ( this.enumDef.data == null )
          return false;

      var n = this.getElementIndex();
      if ( n+1 < this.enumDef.data.length )
      {
          this.setValue( this.enumDef.data[n+1].num, from );
          return true;
      }
      return false;
  }
  /*****/

1470  Class : PrimitiveEnum
      Method: decrement
      Params: Listener from
      Return: Boolean - if changed
      Description: Decrements the enumeration value to the previous value in
      the set definition. Returns false if already at the first element.
      *****/
1475  function _PrimitiveEnum_decrement( from )
  {
      if ( this.enumDef.data == null )
          return false;

      var n = this.getElementIndex();
      if ( n > 0 )
      {
          this.setValue( this.enumDef.data[n-1].num, from );
          return true;
      }
      return false;
  }
  /*****/

1490  Class : PrimitiveEnum
      Method: fromString

```

```

1495      Params: String text
      Return: Number
      Description: converts the string given into a number
      *****/
      function _PrimitiveEnum_fromString(text)
      {
1500          if(text==null){ return null;}
          //parse out the first integer in the string.
          //must begin with an number.
          var number=parseInt(text);
          //if a number was found return it
          if(!isNaN(number)){ return number;}
1505          // else return the unchanged text
          return text;
      }
1510  *****/
      Class : PrimitiveEnum
      Method: getAllowed, hasAllowed and setAllowed
      Params: varies
      Return: varies
1515      Description: a restriction for the PrimitiveEnum is which enums are
          allowed these methods handle this restriction.
      *****/
      function _PrimitiveEnum_getAllowed()
1520  {
          return this.allowed;
      }

1525  function _PrimitiveEnum_hasAllowed()
      {
          return this.allowed != null;
      }

1530  function _PrimitiveEnum_setAllowed(allowed )
      {
          // Allow ANY 'allowed' string since it may now be text too.
          this.allowed=allowed;
      }
1535  *****/
      Class : PrimitiveEnum
      Method: valuedatevalue
      Params: Object test
      Return: none

```

```

1540 Description: Overrides the default validateValue() method.
      If there are no restrictions, then an implicit restriction is that
      it must be one of the number in the set definition
      *****
function _PrimitiveEnum_validateValue( test )
1545 {
    var allowed = this.getAllowed();
    if ( allowed != null )
    {
        // search for allowed value by looking for "3," in "1,2,3,4,5,"
        // note that this will break if spaces are present
        allowed = "," + allowed + ",";
        var searchVal = " " + test + " ";
        if ( allowed.indexOf( searchVal ) == -1 )
        {
            throw new CJException.InvalidValue("Enumerated value \"\" + test +
            \"\" is disallowed, subset is \"\" + this.getAllowed() + "\"\" );
        }
    }
    //check if the test value exists in enumeration
    if ( this.data != null )
    {
        for ( var i=0; i<this.enumDef.data.length; i++ )
        {
            if ( this.enumDef.data[i].num == test )
            {
                this.base_validateValue( test );
                return;
            }
        }
    }
    throw new CJException.InvalidValue(
    "Value \"\" + test + "\" for PrimitiveEnum \"\" + this.id
    + "\" is not in definition" );
}
1575

PrimitiveEnum.prototype = new PrimitiveBase();
PrimitiveEnum.prototype.valueChanged = _PrimitiveEnum_valueChanged;
PrimitiveEnum.prototype.base_validateValue = _PrimitiveBase_validateValue;
1580 PrimitiveEnum.prototype.getSetDefinition = _PrimitiveEnum_getSetDefinition;
PrimitiveEnum.prototype.getElementText = _PrimitiveEnum_getElementText;
PrimitiveEnum.prototype.getElementIndex = _PrimitiveEnum_getElementIndex;
PrimitiveEnum.prototype.increment = _PrimitiveEnum_increment;
PrimitiveEnum.prototype.decrement = _PrimitiveEnum_decrement;
1585 PrimitiveEnum.prototype.fromString = _PrimitiveEnum_fromString;

```

```

PrimitiveEnum.prototype.validateValue
PrimitiveEnum.prototype.allowed
PrimitiveEnum.prototype.getAllowed
PrimitiveEnum.prototype.hasAllowed
1590 PrimitiveEnum.prototype.setAllowed
    = _PrimitiveEnum_validateValue;
    = null;
    = _PrimitiveEnum_getAllowed;
    = _PrimitiveEnum_hasAllowed;
    = _PrimitiveEnum_setAllowed;

1595 /*****
*****
CLASS: PrimitiveOctetString
1600 A PrimitiveOctetString is a sequence of octets (bytes) stored as
    a hexadecimal string. It's data type is String. There are helper
    methods to get and set individual bytes, but it is recommended that
    the content typical be set as a whole.
1605 The class's data type is a String.
Restrictions:
    maxLength - max octet length. This the stored string length may be not
                more that twice this value since there are two characters
                required per byte. (null = no limit)
1610
*****
*****
function PrimitiveOctetString( id, writable, initData, maxLength)
1615 {
    this.setMaxLength(maxLength);
    this.PrimitiveBase( id, writable, initData );
}

1620 /*****
Class : PrimitiveOctetString
Method: getLength
Params:
Return: number of octets in the string
1625 Description: This method returns the number of octets in the string.
    Which is exactly half of the string length.
*****
function _PrimitiveOctetString_getLength( )
{
1630     return ( ( this.data == null ) ? 0 : this.data.length / 2 );
}

```



```

1635 /*****
Class : PrimitiveOctetString
Method: getByte
Params: Number position
Return: Number (byte)
Description: Returns the octet (byte) at the specified byte offset.
Exceptions: OutOfRange
*****/
1640 function _PrimitiveOctetString_getByte( pos )
{
    if ( pos < this.getLength() )
    {
        return parseInt( this.data.substr( pos * 2, 2 ), 16 );
    }
    throw new CJException.OutOfRange(
        "PrimitiveOctetString_getByte - byte position \"\" + pos
        + \"\\\" invalid, range is 0 to \" + this.getLength() + \"-1\" );
1650 }

1655 /*****
Class : PrimitiveOctetString
Method: setByte
Params: Number position, Number octet(byte)
Return: none
Description: Sets the specified octet (byte) at the specified offset.
Exceptions: OutOfRange
*****/
1660 function _PrimitiveOctetString_setByte( pos, theByte, from )
{
    if ( pos < this.getLength() )
    {
        theByte = theByte & 0xff;
        var insert = theByte.toString( 16 );
        {
            if ( insert.length == 1 )
            {
                insert = "0" + insert;
            }
            // this is a pain to replace a segment of a string...
            var tmp = this.data.substring( 0, pos*2 );
            tmp += insert;
            tmp += this.data.substring( pos*2+2, this.getLength()*2 );
            return this.setValue( tmp, from );
        }
    }
1675 }

```

```

1680     throw new CJException.OutOfRange(
        "PrimitiveOctetString_setByte - byte position \"\" + pos
        + \"\\\" invalid, range is 0 to \" + this.getLength() + \"-1\" );
    }

/*****
Class : PrimitiveOctetString
Method: getMaxLength, hasMaxLength and setMaxLength
Params: varies
Return: varies
Description: The restriction for the PrimitiveOctetString is its length
            these methods handle that restriction.
*****/
1690     function _PrimitiveOctetString_getMaxLength()
    {
        if ( this.maxlength == null )
            return Number.MAX_VALUE;
        else
            return this.maxlength;
    }

1695     function _PrimitiveOctetString_hasMaxLength()
    {
        return this.maxlength != null;
    }

1700     function _PrimitiveOctetString_setMaxLength(maxlen)
    {
        if ( maxlen < 0 )
        {
            throw new CJException.InvalidValue(
                "max length \" + maxlen + \" must be >= 0\" );
        }
        this.maxlength=maxlen;
    }

1705

1710

1715 /*****
Class : PrimitiveOctetString
Method: validateValue
Params: OctetString test
Return: null
Description: checks the test value against the primitive's restrictions
*****/
1720     function _PrimitiveOctetString_validateValue( test )
    {

```


1770 order 10 bits is the bacnet object type, and the low order 22 bits
 is the instance number.

 The class's data type is a Number.

```
1775 *****  
function PrimitiveBacnetOID( id, writable, initData )  
{  
  *****/
```

```
    this.PrimitiveBase( id, writable, initData );  
}
```

```
1780 // get object type (high 10 bits)  
function _PrimitiveBacnetOID_getType( )  
{  
  return ( parseInt(this.data) );  
}
```

```
1785 }
```

```
    // get object instance (low 22 bits)  
    function _PrimitiveBacnetOID_getInstance( )  
    {
```

```
1790      return ( parseInt(this.data.substr(this.data.indexOf(":")+1) ) );  
    }
```

```
    // set object type (high 10 bits)  
    function _PrimitiveBacnetOID_setType( type, from )  
    {
```

```
1795      return this.setValue( type + ":" + this.getInstance(), from );  
    }
```

```
    // set object instance (low 22 bits)
```

```
1800 function _PrimitiveBacnetOID_setInstance( instance, from )  
{
```

```
    return this.setValue( this.getType() + ":" + instance, from );  
}
```

```
1805 *****  
Class : PrimitiveBacnetOID  
Method: fromString  
Params: String text  
Return: Number  
Description: converts the string given into a bacnetOID in integer form  
*****/
```

```
1810 function _PrimitiveBacnetOID_fromString(text)  
{  
  if (text==null)
```

```

1815 {
1816     return "-1:-1";
1817 }
1818 //first integer is the type
1819 var type=parseInt(text);
1820 //integer after ':' is the instance
1821 var instance=parseInt(text.substr(text.indexOf(":")+1));
1822 //if both numbers were successfully parsed then return the bacnetOID
1823 if(!isNaN(type) && !isNaN(instance))
1824 {
1825     return text;
1826 }
1827 //throw an exception if the conditions above were not met
1828 throw new CJException.InvalidValue("\nPrimitiveBacnetOID: fromString() must be given a string"+
1829 "\nof the form \"decimal:decimal\". It received \""+text+"\".");
1830 }
1831
1832 /*****
1833 Class : PrimitiveBacnetOID
1834 Method: toString
1835 Params:
1836 Return: String
1837 Description: bacnetOID in String form "type:instance"
1838 *****/
1839 function _PrimitiveBacnetOID_tostring()
1840 {
1841     //There is a little bit of a debate over what this should return "".
1842     //when the value is null. For now we are returning the empty string "".
1843     if(this.data==null)
1844     {
1845         return "-1:-1";
1846     }
1847     else
1848     {
1849         return this.data;
1850     }
1851 }
1852
1853 PrimitiveBacnetOID.prototype = new PrimitiveBase();
1854 PrimitiveBacnetOID.prototype.getType = _PrimitiveBacnetOID_getType;
1855 PrimitiveBacnetOID.prototype.getInstance = _PrimitiveBacnetOID_getInstance;

```

```

1865 PrimitiveBacnetOID.prototype.setType
      = _PrimitiveBacnetOID_setType;
PrimitiveBacnetOID.prototype.setInstance
      = _PrimitiveBacnetOID_setInstance;
PrimitiveBacnetOID.prototype.toString
      = _PrimitiveBacnetOID_toString;
PrimitiveBacnetOID.prototype.fromString
      = _PrimitiveBacnetOID_fromString;

```

```

1870 /*****
      *****/

```

```

1875 CLASS: PrimitiveDate
      A PrimitiveDate is a 4 octet value containing a year, month, day,
      and day of week. The year is based from 1900. Any field may be
      0xFF indicating an unspecified value.

```

```

1880 The class's data type is a Number.
      *****/
      *****/
function PrimitiveDate( id, writable, initData )
1885 {
      this.PrimitiveBase( id, writable, initData );
}

```

```

      // Note - if this.data is 'null', you get 0 for everything
1890 function _PrimitiveDate_getYear()
      { return (( this.data & 0xff000000) >>> 24 ); }

```

```

1895 function _PrimitiveDate_getMonth()
      { return (( this.data & 0x00ff0000) >>> 16 ); }
function _PrimitiveDate_getDay()
      { return (( this.data & 0x0000ff00) >>> 8 ); }

```

```

1900 function _PrimitiveDate_getDayOfWeek()
      { return (( this.data & 0x000000ff) ); }
function _PrimitiveDate_setYear( v, from )
      { this.setValue((this.data & 0x00ffffff) + (v<<24), from); }
1905 function _PrimitiveDate_setMonth( v, from )
      { this.setValue((this.data & 0xff00ffff) + (v<<16), from); }

```

```

1910 function _PrimitiveDate_setDay( v, from )
    { this.setValue((this.data & 0xffff00ff) + (v<< 8), from); }

1915 function _PrimitiveDate_setDayOfWeek( v, from )
    { this.setValue((this.data & 0xffffff00) + (v), from ); }

    // return the primitive's value
1920 function _PrimitiveDate_getValue()
    {
        //if the value appears negative change it to the positive value
        if(this.data<0) return 0x100000000+this.data
        return this.data;
    }

    /*****
    Class : PrimitiveDate
    Method: fromString
    Params: String text
    Return: Number
    Description: Converts the date string given into an integer form.
    Uses the internet standard "GMT" format date as produced by the java
    Date.toGMTString(). The format is "Sat 12 Aug 1995". Note capitalization.
    *****/
1930 function _PrimitiveDate_fromString(text)
    {
        if(text==null){ return null;}
        var dayOfWeek;
        var day;
        var month;
        var year;

        //split the text string at every place there is one or more spaces.
        //this creates an array with all the separated elements
        var date=text.split(/ +/);

        //test for improperly formatted date
        CJAssert(date.length==4,"\\nPrimitiveDate: fromString() must be given a string "+
            "\\nof the form \\\"Sat 12 Aug 1995\\\". It received \\\""+text+"\\", "+
            "\\nwhich has too many spaces.");

        //first element is the day of the week
        for(var loop=0; loop<PrimitiveDate.dayArray.length;loop++)
        {
            if(date[0].toLowerCase()==PrimitiveDate.dayArray[loop].toLowerCase()) break;

```

```

1955 if(loop>=PrimitiveDate.dayArray.length){dayOfWeek==null;}
    else
    {
        if(loop==0){dayOfWeek=0xFF;}
        else{dayOfWeek=loop;}
    }

1960 //second element is the day of the month,
    //replaces every occurrence of "*" with "255" (ie 0xFF) before parsing
    //The internal form of the date uses 0xFF as the wild card value
    day=parseInt(date[1].replace(/\*/g, "255"));
    if(day<0||day>255){day=NaN;}

1965 //third element is the month
    for(loop=0; loop<PrimitiveDate.monthArray.length;loop++)
    {
        if(date[2].toLowerCase()==PrimitiveDate.monthArray[loop].toLowerCase()) break;
    }
    if(loop>=PrimitiveDate.monthArray.length){month==null;}
    else
    {
        if(loop==0){month=0xFF;}
        else{month=loop;}
    }

1975 //fourth element is the year
    //replaces every occurrence of "*" with "2155" (ie 255+1900) before parsing
    //subtract 1900 to get the year to be stored, must be 0-255
    year=parseInt(date[3].replace(/\*/g, "2155"))-1900;
    if(year<0||year>255){year=NaN;}

1980 //throw an exception if any of the elements did not parse correctly
    CJAssert(dayOfWeek!=null || !isNaN(day) || month!=null || !isNaN(year),
        "\nPrimitiveDate: fromString() must be given a string "+
        "\nof the form \"Sat 12 Aug 1995\". It received \""+text+"\\".");

1985
    var data;
    data=(data & 0x00ffffff) + (year<24);
    data=(data & 0xff00ffff) + (month<<16);
    data=(data & 0xffff00ff) + (day<< 8);
    data=(data & 0xfffffff0) + (dayOfWeek);

1995 return data;
    }
/*****

```



```

2000 Class : PrimitivesDate
      Method: toString
      Params:
      Return: String
      Description: Returns the string form of the Date.
                  The form is "www D mmM YYYY" ex. "Sat 12 Aug 1995"
2005 *****/
      function _PrimitivesDate_toString()
      {
          //There is a little bit of a debate over what this should return
          //when the value is null. For now we are returning the empty string "".
          if(this.data==null){return "";}

          //looks up the proper dayOfWeek and month in the respective arrays
          //and gets the day and year. Then concatenates them together in the
          //correct order separated by spaces.
          var dayOfWeek=this.getDayOfWeek();
          var day=this.getDay();
          var month=this.getMonth();
          var year=this.getYear();

          if(dayOfWeek==0xFF){dayOfWeek=0;}
          dayOfWeek=PrimitivesDate.dayArray[dayOfWeek];

          if(month==0xFF){month=0;}
          month=PrimitivesDate.monthArray[month];

          if(day==0xFF){day="*";}

          if(year==0xFF){year="*";}
          else{year=year+1900}

          return (dayOfWeek+" "+day+" "+month+" "+year);
      }

2035 PrimitivesDate.dayArray=new Array("","Mon","Tue","Wed",
      "Thu","Fri","Sat","Sun");
      PrimitivesDate.monthArray=new Array("","Jan","Feb","Mar","Apr","May","Jun",
      "Jul","Aug","Sep","Oct","Nov","Dec");

2040 PrimitivesDate.prototype = new PrimitivesDate();
      PrimitivesDate.prototype.getYear = _PrimitivesDate_getYear;
      PrimitivesDate.prototype.getMonth = _PrimitivesDate_getMonth;
      PrimitivesDate.prototype.getDay = _PrimitivesDate_getDay;
      PrimitivesDate.prototype.getDayOfWeek = _PrimitivesDate_getDayOfWeek;
      PrimitivesDate.prototype.setYear = _PrimitivesDate_setYear;

```

```

2045 PrimitiveDate.prototype.setMonth      = _PrimitiveDate_setMonth;
    PrimitiveDate.prototype.setDay      = _PrimitiveDate_setDay;
    PrimitiveDate.prototype.setDayOfWeek = _PrimitiveDate_setDayOfWeek;
    PrimitiveDate.prototype.toString    = _PrimitiveDate_toString;
    PrimitiveDate.prototype.fromString  = _PrimitiveDate_fromString;
2050 PrimitiveDate.prototype.getValue    = _PrimitiveDate_getValue;
    /*****
    *****/

    CLASS: PrimitiveTime

2055     A PrimitiveTime is a 4 octet value containing a hour, minute, second
        and 100ths of a second. Any field may be 0xFF indicating an
        unspecified value.

2060     The class's data type is a Number.

    *****/
    *****/
    function PrimitiveTime( id, writable, initData )
2065 {
    {
        this.PrimitiveBase( id, writable, initData );
    }

    // Note - if this.data is 'null', you get 0 for everything
2070 function _PrimitiveTime_getHour()
    {
        return ( ( this.data & 0xff000000) >>> 24 ); }

    function _PrimitiveTime_getMinute()
2075 {
        return ( ( this.data & 0x00ff0000) >>> 16 ); }

    function _PrimitiveTime_getSecond()
    {
        return ( ( this.data & 0x0000ff00) >>> 8 ); }

2080 function _PrimitiveTime_get100ths()
    {
        return ( ( this.data & 0x000000ff) ); }

    function _PrimitiveTime_setHour( v, from )
    {
        this.setValue((this.data & 0x00ffffff) + (v<<24), from );}

2085 function _PrimitiveTime_setMinute( v, from )
    {
        this.setValue((this.data & 0xff00ffff) + (v<<16), from );}

    function _PrimitiveTime_setSecond( v, from )
    {
        this.setValue((this.data & 0xffff00ff) + (v<< 8), from );}

2090

```

```

function _PrimitiveTime_set100ths( v, from )
{ this.setValue((this.data & 0xfffff00) + (v), from ); }

// return the primitive's value
function _PrimitiveTime_getValue()
{
    //if the value appears negative change it to the positive value
    if(this.data<0) return 0x100000000+this.data
    return this.data;
}

/*****
Class : PrimitiveTime
Method: fromString
Params: String text
Return: Number
Description: converts the Time string given into an integer form.
             The string is of the form "h:mm:ss.HH",
             where h is hours (one or 2 digits), mm is minutes (2 digits),
             ss is seconds (2 digits), and HH is hundredths (2 digits).
             Any one of these fields can be the single char '*' for wild card.
*****/
function _PrimitiveTime_fromString(text)
{
    if(text==null){ return null;}
    var hour;
    var minute;
    var second;
    var hundredth;

    //replaces every occurrence of "*" with "255" (ie 0xFF)
    //The internal form of the time uses 0xFF as the wild card value
    var time=text.replace(/\*/g, "255")

    //split the text string at every place there is a ":".
    //this creates an array with all the separated elements
    time=time.split(/:/);

    //test for improperly formatted time
    CJAssert(time.length==4,"PrimitiveTime: fromString() must be given a string "+
    "of the form \"HH:MM:SS:hh\". It received \""+text+"\", "+
    "\"\nwhich has too many \":\"'s");

    //first element is the hour
    hour=parseInt(time[0]);
    if(hour<0|hour>255){hour=NaN;}
}

```

```

2140 //second element is the minutes
minute=parseInt(time[1]);
if(minute<0|minute>255){minute=NaN;}

2145 //third element is the seconds
second=parseInt(time[2]);
if(second<0|second>255){second=NaN;}

//fourth element is the get100ths of a second
hundredth=parseInt(time[3]);
if(hundredth<0|hundredth>255){hundredth=NaN;}

2150 //throw an exception if any of the elements did not parse correctly
CJAssert(!isNaN(hour) || !isNaN(minute) || !isNaN(second) || !isNaN(hundredth),
"\nPrimitiveTime: fromStringQ must be given a string "+
"\nof the form \"HH:MM:SS:hh\". It received \""+text+"\".");

2155 var data;
data=(data & 0x00fffff) + (hour<<24);
data=(data & 0xff00ffff) + (minute<<16);
data=(data & 0xffff00ff) + (second<< 8);
data=(data & 0xfffffff0) + (hundredth);

2160 return data;
}
/*****
2165 Class : PrimitiveTime
Method: toString
Params:
Return: String
Description: returns the string form of the Date.
The form is "HH:MM:SS:hh" ex. "23:59:59:99"
uses the 24 hour clock and the minutes, seconds, and 100s
have leading zeros if <10;
*****/
function _PrimitiveTime_toString()
2175 {
//There is a little bit of a debate over what this should return
//when the value is null. For now we are returning the empty string "".
if(this.data==null){return "";}

2180 //gets the time values. If they are 0xFF make them the wildcard "*" otherwise
//if they are only one digit add a leading 0. Hour does not have that.

```

```

2185 //Then concatenate them together in the correct order separated by spaces.
    var hour=this.getHour();
    var minute=this.getMinute();
    var second=this.getSecond();
    var hundredths=this.get100ths();

    if(hour==0xFF){dayOfWeek="*";}
2190     if(minute==0xFF){minute="*";}
    else{minute=minute<10?"0"+minute:minute}

    if(second==0xFF){second="*";}
2195     else{second=second<10?"0"+second:second}

    if(hundredths==0xFF){hundredths="*";}
    else{hundredths=hundredths<10?"0"+hundredths:hundredths}
2200     return (hour+": "+minute+": "+second+": "+hundredths);
    }

    PrimitiveTime.prototype = new PrimitiveBase();
2205     PrimitiveTime.prototype.getHour = _PrimitiveTime_getHour;
    PrimitiveTime.prototype.getMinute = _PrimitiveTime_getMinute;
    PrimitiveTime.prototype.getSecond = _PrimitiveTime_getSecond;
    PrimitiveTime.prototype.get100ths = _PrimitiveTime_get100ths;
    PrimitiveTime.prototype.setHour = _PrimitiveTime_setHour;
    PrimitiveTime.prototype.setMinute = _PrimitiveTime_setMinute;
    PrimitiveTime.prototype.setSecond = _PrimitiveTime_setSecond;
    PrimitiveTime.prototype.set100ths = _PrimitiveTime_set100ths;
    PrimitiveTime.prototype.toString = _PrimitiveTime_toString;
    PrimitiveTime.prototype.fromString = _PrimitiveTime_fromString;
2215     PrimitiveTime.prototype.getValue = _PrimitiveTime_getValue;

    /*****
    CLASS: PrimitiveBitString
    The PrimitiveBitString is an array of bit values stored as a string,
    of ones and zeros, e.g. "011010010010". Most significant bit is first.
    The class's data type is a String.
2220
2225

```

```

2230 Restrictions:
      maxbits - max # of bits allowed in bit string (no value = infinite)
      allowed - string of '1' and '0' characters where '1' indicates that the
                  bit position is allowed to be set. This restriction may be
                  'null', indicating that there is no restriction on which bits
                  may be set. (null=any)
2235 *****
      *****
      *****
      function PrimitiveBitString( id, writable, initData, defID, maxbits, allowed )
      {
          this.setMaxBits(maxbits);
          this.setAllowed(allowed);
          if ( defID != null )
          {
              this.enumDef = PrimitiveBase.findPrimitive( defID );
              this.enumDef.addListener( this );
          }
          this.PrimitiveBase( id, writable, initData );
      }
2240
2245
2250 /*****
      Class : PrimitiveBitString
      Method: valueChanged
      Params: prim
      Return: none
2255 Description: A PrimitiveBitString is a listener of it's
                  PrimitiveSetDefinition. Should the definition set change, then
                  the representation of the bit values should also change. This does
                  nothing more than propagate a change to the prim's listeners.
                  *****
2260 function _PrimitiveBitString_valueChanged( changedDef )
      {
          // propagate change
          this.notifyListeners( this );
      }
2265
      /*****
      Class : PrimitiveBitString
      Method: getLength
      Params:
2270 Return: Number of bits
      Description: Returns the number of defined bits in the bit string.
      *****
      function _PrimitiveBitString_getLength( )

```

```

2275 {
    if ( this.data == null ){return 0;}
    else{return this.data.length;}
}

2280 /*****
Class : PrimitiveBitString
Method: getBit
Params: Number pos
Return: Boolean isSet
Description: Returns if the boolean value of the specified bit
position.
Exceptions: OutOfRange
*****/
function _PrimitiveBitString_getBit( pos )
2290 {
    if ( pos < this.getLength() )
    {
        bitValue=this.data.charAt(pos);
        if(bitValue=="1"){return true;}
        if(bitValue=="0"){return false;}
        throw new CJException.InvalidValue(
2295 "PrimitiveBitString_getBit - bit position \"\" + pos
+ \"\\\" invalid, must be 0 or 1 found \"\"+bitValue );
    }
    throw new CJException.OutOfRange(
2300 "PrimitiveBitString_getBit - bit position \"\" + pos
+ \"\\\" invalid, range is 0 to \" + this.getLength() + "-1" );
}

2305 // get the bit value at the specified position
/*****
Class : PrimitiveBitString
Method: setBit
Params: Number pos, Boolean val
Return: none
Description: Set's the bit value at the specified position
Exceptions: OutOfRange & InvalidValue
*****/
function _PrimitiveBitString_setBit( pos, val, from )
2315 {
    if ( pos < this.getLength() )
    {
        if( pos > this.getMaxBits() )
        {
            throw new CJException.InvalidValue(
2320

```

```

        "PrimitiveBitstring_setBit - at bit position \" + pos
        + "\" is exceeds maximum bit restriction");
    }
    if(this.hasAllowed())
    {
        if(this.getAllowed().charAt(pos) != "1")
        {
            throw new CJException.InvalidValue(
                "PrimitiveBitstring_setBit - at bit position \" + pos
                + "\" is restricted");
        }
    }
    // if the value is 0, false, "false, or null set to false
    // if it is anything else set to true
    if ( val == 0 || val == false || val=="false" || val == null )
    {
        bitValue = "0";
    }
    else
    {
        bitValue = "1";
    }
    var newString = null;
    // if setting the position of the first bit
    if (pos==0)
    {
        newString = bitValue+this.data.substr(pos+1);
    }
    //if setting the position of the last bit
    else if (pos==this.getLength())
    {
        newString = this.data.slice(0, pos)+bitValue;
    }
    else
    {
        //if setting the position of any other bit
        newString = this.data.slice(0, pos)+bitValue+this.data.substr(pos+1);
    }
    this.setValue( newString, from );
    return newString;
}
throw new CJException.OutOfRange(
    "PrimitiveBitstring_setBit - bit position \" + pos
    + "\" invalid, range is 0 to " + this.getLength() + "-1" );
2365 }

```



```

2370 // access the set definition
function _PrimitiveBitString_getSetDefinition( )
{
    return this.enumDef;
}

2375 // get the element text for the specified bit position.
function _PrimitiveBitString_getElementText( bit )
{
    return this.enumDef.getElementText( bit );
}

2380 /*****
Class : PrimitiveBitString
Method: getMaxBits, hasMaxBits and setMaxBits
Params: varies
Return: varies
Description: a restriction for the PrimitiveBitString is the maximum
number of bits these methods handle this restriction.
*****/
function _PrimitiveBitString_getMaxBits()
{
    if ( this.maxbits == null )
        return Number.MAX_VALUE;
    else
        return this.maxbits;
}

2395 function _PrimitiveBitString_hasMaxBits()
{
    return this.maxbits != null;
}

2400 function _PrimitiveBitString_setMaxBits( max )
{
    if ( max < 0 )
    {
        throw new CJException.InvalidValue("MaxBits " + max + " must be >= 0" );
    }
    this.maxbits = max;
}

2410 /*****
Class : PrimitiveBitString
Method: getAllowd, hasAllowed and setAllowed
*****/

```

```

Params: varies
Return: varies
2415 Description: a restriction for the PrimitiveBitString is which bits are
        allowed to be set these methods handle this restriction.
        *****
function _PrimitiveBitString_getAllowed()
{
2420     return this.allowed;
}

function _PrimitiveBitString_hasAllowed()
{
2425     return this.allowed != null;
}

function _PrimitiveBitString_setAllowed( allowed )
{
2430     this.allowed = allowed;
}
/*****
Class : PrimitiveBitString
Method: validateValue
Params: BitString test
Return: null
Description: checks the test value against the primitive's restrictions
*****
function _PrimitiveBitString_validateValue( test )
{
2440     if ( test.length > this.getMaxBits() )
    {
        throw new CJException.InvalidValue("BitString of " + test.length +
            " bits exceeds limit of " + this.getMaxBits() + " bits");
    }
2445     // get and save our "allow bit positions" - subset of total
    var allowed = this.getAllowed();

    // go over each bit and check it
    for (var i=0; i<test.length; i++)
    {
        var c = test.charAt(i);

        // ensure it contains only '0' and '1' chars
        if ( c != '1' && c != '0' )
        {
2455             throw new CJException.InvalidValue("PrimitiveBitString: Invalid BitString \"\"
                + test + "\" may contain only '0' and '1'\\n");
        }
    }
}

```

```

2460     }
    // ensure that only allowed bits, if any, are set
    if ( allowed != null && c == '1' &&
        ( i >= allowed.length || allowed.charAt(i) != '1' ))
    {
2465         throw new CJException.InvalidValue("BitString \"\" + test +
            "\" contains disallowed bits, mask is \"\" + allowed + "\"\\n\"");
    }

    // ensure that an "on" bit is in our definition (if we have one)
    if ( c == '1' && this.enumDef != null )
    {
        try
        {
2470             this.enumDef.getElementText(i)
        }
        catch ( ex )
        {
2475             if(ex instanceof CJException.NotFound)
                throw new CJException.InvalidValue("BitString \"\" + test +
                    "\" contains undefined bit at offset \" + i + "\"\\n\"");
            else
                throw ex;
        }
    }
2480
2485 }

PrimitiveBitString.prototype = new PrimitiveBase();
2490 PrimitiveBitString.prototype.valueChanged = _PrimitiveBitString_valueChanged;
PrimitiveBitString.prototype.getLength = _PrimitiveBitString_getLength;
PrimitiveBitString.prototype.getSetDefinition = _PrimitiveBitString_getSetDefinition;
PrimitiveBitString.prototype.getBit = _PrimitiveBitString_getBit;
PrimitiveBitString.prototype.setBit = _PrimitiveBitString_setBit;
2495 PrimitiveBitString.prototype.getElementText = _PrimitiveBitString_getElementText;
PrimitiveBitString.prototype.maxbits = null;
PrimitiveBitString.prototype.getMaxBits = _PrimitiveBitString_getMaxBits;
PrimitiveBitString.prototype.hasMaxBits = _PrimitiveBitString_hasMaxBits;
PrimitiveBitString.prototype.setMaxBits = _PrimitiveBitString_setMaxBits;
2500 PrimitiveBitString.prototype.allowed = null;
PrimitiveBitString.prototype.getAllowed = _PrimitiveBitString_getAllowed;
PrimitiveBitString.prototype.hasAllowed = _PrimitiveBitString_hasAllowed;
PrimitiveBitString.prototype.setAllowed = _PrimitiveBitString_setAllowed;
PrimitiveBitString.prototype.validateValue = _PrimitiveBitString_validateValue;

```

2505

2510 /*****

CLASS: PrimitiveExpr

2515 A PrimitiveExpr is a data value that is the result of an
arbitrary JavaScript expression. If is constructed with
initial data that is the expression to be evaluated, and
a list of primitive IDs that are the other primitives that
the expression depends on. If any of the dependent primitives
are undefined, then the current value is undefined. Once
all of the primitives are defined, then the expression may
and will be evaluated.

2525 Being writable implies that the *expression* may be redefined.

It is illegal to invoke setValue()/setMasterValue() on this.
The master value is always 'null'. Therefore calling isChanged()
is rather meaningless.

2530 To change the expression, use setExpression()

The class's data type is ANY.

2535 *****/
function PrimitiveExpr(id, writable, initExpr, deplist)

{
this.expression = initExpr;
this.dependents = null;
this.setDependents(deplist);

this.PrimitiveBase(id, writable, null);

this.evaluate();

2545 }

/*****

Class : PrimitiveExpr

Method: setExpression

Params: String newExpr new JavaScript expression

2550

```

                Array deplList      Stringarray of primitive IDs (or null if none)
Return: none
Description: Sets a new expression and dependency list
Exceptions: none
*****
2555 function _PrimitiveExpr_setExpression( newExpr, deplList )
{
    if ( this.iswritable() )
    {
        2560         this.expression = newExpr;
        this.setDependents( deplList );
        this.evaluate();
    }
    else
    {
        2565         throw new CJException.ReadOnly(
            "Cannot set expression for primitive \"\" + this.id + "\"\" );
        }
    }
}
2570 /*****
Class : PrimitiveExpr
Method: valueChanged
Params: none
Return: none
2575 Description: The valueChanged callback is invoked whenever a dependent
primitive is changed. The PrimitiveExpr primitive will register
itself as a listener of the dependent primitives.
Exceptions: none
*****
2580 function _PrimitiveExpr_valueChanged( )
{
    this.evaluate();
}
2585 /*****
Class : PrimitiveExpr
Method: evaluate
Params: none
Return: none
2590 Description: Evaluates the expression. If any of the dependent
primitives are undefined, then it is assumed that the expression
be 'null'. Otherwise the result of the expression is stored
2595 in the current value.

```

```

Exceptions: none
*****
function _PrimitiveExpr_evaluate( )
*****
2600 {
    var newVal = null;
    var isUndefined = false;

    // check to see if any dependents are undefined
    if ( this.dependents != null )
    {
        for ( var i=0; i<this.dependents.length; i++ )
        {
            if ( this.dependents[i].isUndefined() )
            {
                isUndefined = true;
                break;
            }
        }
    }
    // evaluate only if not undefined
    if ( ! isUndefined )
    {
        try
        {
            newVal = eval( this.expression );
        }
        catch ( e )
        {
            throw new CJException.InvalidValue(
                "PrimitiveExpr - cannot evaluate expression for primitive \"\" +
                this.getId() + "\"\"\\n\" + e.toString() );
        }
    }

    // test for change
    if ( this.data != newVal )
    {
        // update value and notify
        this.data = newVal;
        this.notifyListeners( this );
        PrimitiveBase.notifyUIListeners( this, this );
    }
}
2640 /*****
class : PrimitiveExpr
*****

```

```

Method: setDependents (PRIVATE HELPER)
Params: Array deplList    Stringarray of primitive IDs (or null if none)
Return: none
Description: Unregisters from previously listened to primitives, then
             finds all of the new primitives and registers as a listener to them.
Exceptions: none
*****
2645 function _PrimitiveExpr_setDependents( deplList )
    {
        if ( this.dependents != null )
        {
            for ( var i=0; i<this.dependents.length; i++ )
            {
                this.dependents[i].removeListener( this );
            }
            this.dependents = null;
        }
        if ( deplList != null && deplList.length > 0 )
        {
            this.dependents = new Array();
            for ( var i=0; i<deplList.length; i++ )
            {
                this.dependents[i] = PrimitiveBase.findPrimitive( deplList[i] );
                this.dependents[i].addListener( this );
            }
        }
    }
2670 }

// INVALID FUNCTION - NOT ALLOWED - throw exception
function _PrimitiveExpr_setValue()
{
    2675 throw new CJException.InvalidValue(
        "PrimitiveExpr - setValue() is not allowed for primitive \"" +
        this.getId() + "\"");
}

2680 // INVALID FUNCTION - NOT ALLOWED - throw exception
function _PrimitiveExpr_setMasterValue()
{
    throw new CJException.InvalidValue(
        "PrimitiveExpr - setMasterValue() is not allowed for primitive \"" +
        this.getId() + "\"");
}
2685 }

```

```

2690 PrimitiveExpr.prototype = new PrimitiveBase();
    PrimitiveExpr.prototype.setExpression = _PrimitiveExpr_setExpression;
    PrimitiveExpr.prototype.evaluate = _PrimitiveExpr_evaluate;
    PrimitiveExpr.prototype.valueChanged = _PrimitiveExpr_valueChanged;
    PrimitiveExpr.prototype.setDependents = _PrimitiveExpr_setDependents;
    PrimitiveExpr.prototype.setValue = _PrimitiveExpr_setValue;
    PrimitiveExpr.prototype.setMasterValue = _PrimitiveExpr_setMasterValue;

2700 /*****
*****/

    CLASS: PrimitiveTable

2705 PrimitiveTable consists of an array of PrimTableRows. PrimTableRows
    have an id and an array of PrimTableCells. A PrimTableCell has a
    display string and a value string. The display string is used to
    represent the cell in a control and the value string is used when
    communicating with core.

2710 add and removeRow are used to manipulate the table.

2715 *****/
*****/

    function PrimTableCell(value)
    {
        this.value = value;
    }

    function _PrimTableCell_getcopy()
    {
        return (new PrimTableCell(this.value));
    }

    function _PrimTableCell_tostring()
    {
        var text;
        var ch;
        text = "";
        for (var i = 0; i < this.value.length; i++)

```



```

2735     {
2736         ch = this.value.charAt(i);
2737         if ((ch == PrimitiveTable.FORMAT_CELLDATA_SEP) ||
2738             (ch == PrimitiveTable.FORMAT_CELLDATA_END) ||
2739             (ch == PrimitiveTable.FORMAT_ESCAPE_CHAR))
2740         {
2741             text += PrimitiveTable.FORMAT_ESCAPE_CHAR;
2742         }
2743         text += ch;
2744     }
2745     return (text);
2746 }
2747
2748 PrimitiveCell.prototype.getCopy = _PrimitiveCell_GetCopy;
2749 PrimitiveCell.prototype.toString = _PrimitiveCell_ToString;
2750
2751 // note that the cells param is an array of PrimitiveCells
2752 function PrimitiveRow(id, setDefId, cells, userAdded)
2753 {
2754     this.id = id;
2755     this.setDefId = setDefId;
2756     this.cells = cells;
2757     this.userAdded = userAdded;
2758     this.changed = false;
2759     this.deleted = false;
2760 }
2761
2762 function _PrimitiveRow_GetCopy()
2763 {
2764     var newCells;
2765     var newRow;
2766     newCells = new Array();
2767     for (var i = 0; i < this.cells.length; i++)
2768     {
2769         newCells[i] = null;
2770         if (this.cells[i] != null)
2771         {

```

```

        newCells[i] = this.cells[i].getCopy();
    }
}

2785     newRow = new PrimTableRow(this.id, this.setDefId, newCells, this.userAdded);
        newRow.changed = this.changed;
        newRow.deleted = this.deleted;

2790     return (newRow);
}

PrimTableRow.prototype.getCopy = _PrimTableRow_GetCopy;

2795 /*****
Class : PrimitiveTable
Method: PrimitiveTable (constructor)
Params: id - unique id for primitive
        writable - if false add/removeRow will fail
        initData - Array of PrimTableRows
        local - true if the primitive is local to a page and should not
                be submitted after a change
Description: Note that initData is not a string, it is a literal array
of objects.
*****/
2800     function PrimitiveTable(id, writable, initData, numberOfColumns, local)
    {
        this.PrimitiveBase(id, writable, initData);

        this.numberOfColumns = numberOfColumns;

        if (local != null)
        {
            this.setPageLocal(local);
        }
    }

2820 /*****
Class : PrimitiveTable
Method: addRow
Params: Object from
Return: Nothing.
Description: Adds a row to the table.
*****/
2825

```

```

*****/
function _PrimitiveTable_AddRow(from, rowId, setDefId, cellsData)
2830 {
    var limit = this.data.length;
    var cellsArray;

    if (rowId == null)
2835 {
        rowId = PrimitiveTable.ID_NEW;
    }

    if (setDefId == null)
2840 {
        setDefId = "0";
    }

    if (cellsData == null)
2845 {
        cellsArray = new Array();

        for (var index = 0; index < this.numberOfColumns; index++)
2850 {
            cellsArray[index] = new PrimitiveTableCell("");
        }
    }
    else
2855 {
        cellsArray = cellsData;
    }

    this.data[limit] = new PrimitiveTableRow(rowId, setDefId, cellsArray, true);
    this.data[limit].changed = true;
2860
    this.changed = true;

    // Notify the listeners.
    this.notifyListeners(from);
2865
    if (!this.pageLocal)
    {
        PrimitiveBase.notifyUIListeners(this, from);
    }
2870 }

/*****

```

```

2875      Class : PrimitiveTable
      Method: clearChanged
      Params:
      Return:
      Description: Called to set the changed flag for the table to false.
      *****/

2880 function _PrimitiveTable_ClearChanged()
    {
        var newData;
        var index;

2885         this.changed = false;

        // Remove the deleted rows.
        newData = new Array();
        index = 0;

2890         for (var i = 0; i < this.data.length; i++)
        {
            if (this.data[i].deleted != true)
            {
                newData[index] = this.data[i];
                newData[index].deleted = false;
                newData[index].changed = false;
                newData[index].userAdded = false;
                this.data[i] = null;
                ++index;
            }
        }

2900         this.data = newData;

2905         // Set the original data to this one too.
        this.origData = new Array();

2910         for (var i = 0; i < this.data.length; i++)
        {
            this.origData[i] = this.data[i].getCopy();
        }

2915     }
}

/*****
Class : PrimitiveTable
*****/

```

```

Method: deleteRow
Params: Number rowIndex
Return: Object from
Description: Deletes a row from the table.
*****
2920
2925 function _PrimitiveTable_DeleteRow(rowIndex, from)
    {
        var realIndex;
2930         realIndex = this.getRealIndex(rowIndex);
        // THROW EXCEPTION HERE
        CAssert(realIndex != null, "Requested unknown row");
2935         this.data[realIndex].deleted = true;
        this.data[realIndex].changed = true;
        this.changed = true;
2940         // Notify the listeners.
        this.notifyListeners(from);
        if (!this.pageLocal)
        {
2945             PrimitiveBase.notifyUIListeners(this, from);
        }
    }
}
2950 /*****
Class : PrimitiveTable
Method: fromString
Params: String text
Return: Array of PrimTableRows.
Description: Converts the string given into a PrimitiveTable form.
*****
2955
function _PrimitiveTable_FromString(text)
{
2960     var data = null;
    var ch;
    var state;
    var rowId;
    var setDefId;

```

```

2965     var value;
2966     var ignoreCode;
2967     var cellsArray;
2968     var userAdded = false;

2970     var STATE_GETTING_ROW = 0;
2971     var STATE_GETTING_ROWID = 1;
2972     var STATE_GETTING_SETDEFID = 2;
2973     var STATE_GETTING_VALUE = 3;

2975     // Check for the special code to add a new row.
2976     if (text.indexOf(PrimitiveTable.ID_NEW) == 0)
2977     {
2978         // Get a copy of the current data.
2979         data = new Array();
2980         for (var i = 0; i < this.data.length; i++)
2981         {
2982             data[i] = this.data[i].getCopy();
2983         }

2985         text = text.substring(PrimitiveTable.ID_NEW.length);
2986         userAdded = true;
2987     }

2990     if (data == null)
2991     {
2992         data = new Array();
2993     }

2995     state = STATE_GETTING_ROW;
2996     ignoreCode = false;

3000     for (var i = 0; i < text.length; i++)
3001     {
3002         ch = text.charAt(i);

3003         switch (state)
3004         {
3005             case STATE_GETTING_ROW:
3006                 if (ch != PrimitiveTable.FORMAT_ROWINDICATOR)
3007                 {
3008                     throw new CJException.InvalidClass("_PrimitiveTable_FromString - input text \" + text + "\"
3009                     expecting to find a '.*' t
3010                 }

```

```

3010         rowId = "";
3011         setDefId = "";
3012         state = STATE_GETTING_ROWID;
3013         break;
3014     case STATE_GETTING_ROWID:
3015         if (ch == PrimitiveTable.FORMAT_ROWID_SEP)
3016         {
3017             state = STATE_GETTING_SETDEFID;
3018         }
3019         else
3020         {
3021             rowId += ch;
3022         }
3023         break;
3024     case STATE_GETTING_SETDEFID:
3025         if (ch == PrimitiveTable.FORMAT_CELLDATA_START)
3026         {
3027             cellArray = new Array();
3028             data[data.length] = new PrimitiveTableRow(rowId, setDefId, cellArray, userAdded);
3029             value = "";
3030             state = STATE_GETTING_VALUE;
3031         }
3032         else
3033         {
3034             setDefId += ch;
3035         }
3036         break;
3037     case STATE_GETTING_VALUE:
3038         if (ch == PrimitiveTable.FORMAT_ESCAPE_CHAR)
3039         {
3040             ignoreCode = true;
3041         }
3042         else
3043         {
3044             if (ignoreCode == false)
3045             {

```

```

3060         if (ch == PrimitiveTable.FORMAT_CELLDATA_END)
3061         {
3062             cellArray[cellArray.length] = new PrimitiveTableCell(value);
3063             value = "";
3064             if (i < (text.length - 1))
3065             {
3066                 if (text.charAt(i + 1) == PrimitiveTable.FORMAT_ROWINDICATOR)
3067                 {
3068                     state = STATE_GETTING_ROW;
3069                 }
3070                 else
3071                 {
3072                     state = STATE_GETTING_VALUE;
3073                 }
3074             }
3075             else
3076             {
3077                 value += ch;
3078             }
3079             else
3080             {
3081                 value += ch;
3082                 ignoreCode = false;
3083             }
3084             break;
3085         }
3086         default:
3087             throw new CJException.InvalidClass("_PrimitiveTable_FromString - invalid state");
3088             break;
3089     }
3090 }
3091 return (data);
3092 }
3093 }
3094 }
3095 }
3096 }
3097 }
3098 }
3099 }
3100 }
3101 /*****

```



```

Class : PrimitiveTable
Method: getColumn
Params: Number      rowIndex.
        Number      column.
Return: PrimitiveCell columnData.
Description: Sets the column to the stuff specified.
*****
3105 *****/

function _PrimitiveTable_GetColumn(rowIndex, column)
{
    realIndex = this.getRealIndex(rowIndex);

    CJAssert(realIndex != null, "Requested unknown row");
    3115 CJAssert(this.data[realIndex].cells[column] != null, "Requested unknown column");

    return (this.data[realIndex].cells[column]);
}

3120 /*****
Class : PrimitiveTable
Method: getNumberOfRows
Params:
Return: PrimitiveRow
Description: Returns the number of rows in the table.
*****
3125 *****/

function _PrimitiveTable_GetNumberOfRows()
{
    3130 var count = 0;

    if (this.data == null)
    {
        3135 return (0);
    }

    for (var i = 0; i < this.data.length; i++)
    {
        3140 if (this.data[i].deleted != true)
        {
            ++count;
        }
    }

    3145 return (count);
}

```

```

/*****
Class : PrimitiveTable
Method: getRow
Params: rowId - Row to return.
Return: PrimitiveTableRow
Description: Returns a row specified by the index given.
*****/
3150
3155 function _PrimitiveTable_GetRow(rowIndex)
{
    var realIndex;

3160     realIndex = this.getRealIndex(rowIndex);

    // THROW EXCEPTION HERE
    CjAssert(realIndex != null, "Requested unknown row");

3165     return (this.data[realIndex]);
}

/*****
Class : PrimitiveTable
Method: getRowId
Params: rowIndex - Row to return.
Return: id
Description: Returns the row id of a row index.
*****/
3170
3175 function _PrimitiveTable_GetRowId(rowIndex)
{
    var realIndex;

3180     realIndex = this.getRealIndex(rowIndex);

    // THROW EXCEPTION HERE
    CjAssert(realIndex != null, "Requested unknown row");

3185     return (this.data[realIndex].id);
}

3190 /*****
Class : PrimitiveTable
Method: getSetDefId

```

```

3195 Params: rowIndex - Row to return.
      Return: id
      Description: Returns the row id of a row index.
      *****/
3200 function _PrimitiveTable_GetSetDefId(rowIndex)
      {
          var realIndex;

          realIndex = this.getRealIndex(rowIndex);

3205 // THROW EXCEPTION HERE
          CJAssert(realIndex != null, "Requested unknown row");

          return (this.data[realIndex].setDefId);
3210 }

3215 /*****
      Class : PrimitiveTable
      Method: GetUndoAddString
      Params:
      Return: String
      Description: Returns the string format that should be sent to core to undo
                  any new rows that were added by the user.
      *****/
3220 function _PrimitiveTable_GetUndoAddString()
      {
          var text = "";

3225 for (var index = 0; index < this.data.length; index++)
          {
              if (this.data[index].userAdded == true)
              {
                  text += PrimitiveTable.FORMAT_ROWINDICATOR;
                  text += PrimitiveTable.ID_DELETED + this.data[index].id;
                  text += PrimitiveTable.FORMAT_ROWID_SEP;
                  text += this.data[index].setDefId;
                  text += PrimitiveTable.FORMAT_CELLDATA_START;

3235 for (var cell = 0; cell < this.data[index].cells.length; cell++)
                  {
                      text += this.data[index].cells[cell].toString();
                      text += PrimitiveTable.FORMAT_CELLDATA_END;

```

```

3240     }
    }
    return (text);
3245 }

/*****
3250 Class : PrimitiveTable
Method: ifUserAdded
Params:
Return: Boolean
Description: Returns a true if the user has added any rows before any changes
are posted to core.
*****/
3255 *****/

function _PrimitiveTable_IfUserAdded()
{
    for (var i = 0; i < this.data.length; i++)
    {
        if (this.data[i].userAdded == true)
        {
            return (true);
        }
    }
    return (false);
}
3270

/*****
3275 Class : PrimitiveTable
Method: setColumn
Params: Number
        Number
        PrimitiveCell
        Boolean
        rowIndex
        column
        columnData
        notifyUIListeners.
Return:
3280 Description: Sets the column to the stuff specified.
*****/
function _PrimitiveTable_setColumn(from, rowIndex, column, columnData, notifyUIListeners)
{
    realIndex = this.getRealIndex(rowIndex);
3285

```

```

3290 CJAssert(realIndex != null, "Requested unknown row");
    CJAssert(this.data[realIndex].cells[column] != null, "Requested unknown column");

    this.data[realIndex].cells[column] = columnData;
    this.data[realIndex].changed = true;
    this.data[realIndex].deleted = false;

    this.changed = true;

    // Notify the listeners.
    this.notifyListeners( from );

    if ((!(this.pageLocal) && (notifyUIListeners))
    {
        PrimitiveBase.notifyUIListeners(this, null);
    }
    }

3305 /*****
    Class : PrimitiveBase
    Method: setMasterValue
    Params: object newValue
    Return: none
    Description: Sets the master value of the primitive unconditionally
                  and the current value only if never set by the user.
                  It then notifies all listeners of the change.
    Overrides the base class.
    *****/
3315 function _PrimitiveTable_setMasterValue(newData)
    {
        var changed = false;

        if (newData.indexOf(PrimitiveTable.ID_NEW) == 0)
        {
            changed = true;
        }

        newData = this.fromString(newData);

        if (changed == true)
        {

```

```

3335     this.data = newData;
        this.changed = true;
        this.notifyListeners();
    }
    else
    {
        this.origData = newData;
        if (!this.changed)
        {
            this.resetData();
            this.changed = changed;
            this.notifyListeners();
        }
    }
    if ((this.changed == true) && (!this.pagedLocal))
    {
        PrimitiveBase.notifyUIListeners(this, null);
    }
}

3340
3345
3350
3355

/*****
Class : PrimitiveTable
Method: toString
Return: String
Description: "*id;setDefid:value;value;*id:....etc*"
            *id indicates the beginning of a new row and id is
            the id of the row.
            It will only include the rows that have changed.
*****/
function _PrimitiveTable_ToString()
{
    var text = "";
    for (var index = 0; index < this.data.length; index++)
    {
        if (this.data[index].changed == true)
        {
            text += PrimitiveTable.FORMAT_ROWINDICATOR;

```

```

3380         if (this.data[index].deleted == true)
        {
            text += PrimitiveTable.ID_DELETED;
        }
        text += this.data[index].id;
        text += PrimitiveTable.FORMAT_ROWID_SEP;
        text += this.data[index].setDefid;
        text += PrimitiveTable.FORMAT_CELLDATA_START;

3385         for (var cell = 0; cell < this.data[index].cells.length; cell++)
        {
            text += this.data[index].cells[cell].toString();
            text += PrimitiveTable.FORMAT_CELLDATA_END;
        }
    }
}

3395     return (text);
}

3400     function _PrimitiveTable_resetData()
    {
        if (this.origData == null)
        {
3405             this.data = null;
        }
        else
        {
            this.data = new Array();

3410             for (var i = 0; i < this.origData.length; i++)
            {
                this.data[i] = this.origData[i].getCopy();
            }
        }
    }

3415 }

/*****
Class : PrimitiveTable
Method: getRealIndex
Params:
Return: Number or null if not valid.
*****/

```



```
3470 PrimitiveTable.FORMAT_ROWINDICATOR = '*';
    PrimitiveTable.FORMAT_ROWID_SEP = ',';
    PrimitiveTable.FORMAT_CELLDATA_START = ':';
    PrimitiveTable.FORMAT_CELLDATA_SEP = ',';
    PrimitiveTable.FORMAT_CELLDATA_END = ';';
3475 PrimitiveTable.FORMAT_ESCAPE_CHAR = '\\';

    PrimitiveTable.CODE_SELECTED = '$ROW_SELECTED$';

    PrimitiveTable.ID_DELETED = '$ROW_DELETED$';
3480 PrimitiveTable.ID_NEW = '$ROW_ADDNEW$';
```

Exhibit 9

```

1 /*****
    AUTOMATED LOGIC CORPORATION
    Copyright (c) 1999 - 2000 All Rights Reserved
    This document contains confidential/proprietary information.
    *****/
5 @(#)WidgetTextInput.js
    Initial Author: Jide Omisore

10 Description: Text Input Box widget. This will be used for many controls.

    Dependencies:
        /util/browserapi.js
        /util/cjelementhelper.js
        /util/compat.js
        /controls/CJEvent.js
        /controls/exception.js
        /controls/iconhelper.js

20 $Log: /green/webroot/common/controls/widgetTextInput.js $
    * 29 12/01/99 2:25p sappling
    * changed setsize to be based on font size
    *
25 * 28 11/30/99 11:29a Ddavidson
    * fixed focusText bug where is always triggered a change
    *
    * 27 11/29/99 3:14p Ddavidson
    * added focus() methods to focus to (first) text field
30 *
    * 26 11/18/99 3:17p Ddavidson
    * added focusText member to deal with IE not calling onChange when
    * clicking outside of the IFRAME.
    *
35 * 25 9/27/99 9:34a Jomisore
    * Fixed a bug in the setEditable method.
    *
    * 24 9/22/99 3:40p Jomisore
    * Made changes to the style stuff so that things run faster.
40 *
    * 23 9/22/99 12:01p Sappling
    * This has been changed to partially implement the new styles
    *
45 * 22 9/09/99 9:46a Jomisore
    * Added the "invalidatevalue" method
    *
    * 21 9/08/99 4:35p Jomisore
    * Fixed some comments, cleaned up code for code review.
    *
50 * 20 8/16/99 2:23p Jomisore

```

```

* Changed it so that when the widget control is disabled it doesn't set
* the disabled property of the html element to true.
*
* 19 8/13/99 12:31p Jomisoire
55 * Added workaround code for the IE crash bug.
*
* 18 8/10/99 9:39a Jomisoire
* Fixed a problem with the master value in the text input box not being
* reset in some cases.
60 *
* 17 8/09/99 9:17a Jomisoire
* Added a TextDisplayWidget.
*
* 16 8/06/99 11:45a Jomisoire
65 * Changed it so that if size is specified as 0 in the constructor, it
* will automatically adjust the size of the input box to be the number of
* characters+1.
*
* 15 8/04/99 6:02p Jomisoire
70 * Added new style management.
*
* 14 8/04/99 2:06p Jomisoire
* Fixed setValue so that it can accept nulls.
*
* 13 8/04/99 11:49a Jomisoire
75 * Passed "this" as a parameter to the widget callbacks.
*
* 12 8/04/99 10:34a Jomisoire
* Added some more comments.
80 *
* 11 8/03/99 4:49p Jomisoire
* Added category for distinguishing between styles.
*
* 10 8/02/99 2:02p Jomisoire
85 * Changed the names of the owner callbacks used for handling events.
* Swapped the priority of the styles.
*
* 9 7/29/99 5:25p Jomisoire
* Line fix.
90 *
* 8 7/29/99 10:09a Jomisoire
* Fixed the tabIndex to -1 if the widget is not writable or is disabled.
*
* 7 7/28/99 12:40p Jomisoire
95 * Fixed the style adapter that is set when the widget is made not
* writable.
*
* 6 7/28/99 10:46a Jomisoire
* Added the affectStyleById method.
100 *

```

```

* 5 7/28/99 9:15a Jomimore
* Fixed the CjAsserts and cleaned up some code.
*
* 4 7/23/99 6:30p Jomimore
* Added comments.
105 *
* 3 7/23/99 6:26p Jomimore
* Fixed it to use the style manager and added more functionality to it.
*
110 * 1 7/16/99 1:03p Jomimore
* Renamed from TextInputBox.
*
* 3 7/15/99 6:32p Jomimore
* Oops i broke the keypress validation.
115 *
* 1 7/15/99 2:53p Jomimore
* Text input box widget and validation.
*
* 1 07/14/99 3:28p jomimore
120 * Initial implementation
*****/

125 /*****
Class : widgetTextInput

Public methods:
    applyOnBlurStyle
    applyOnFocusStyle
    getDisabled
    getEditable
    getValue
    isValueValid
    setDisabled
    setEditable
    setListener
    setSize
    setStyleState
    setValue
    toElement

130
135
140
145 *****/

/*****
Class : widgetTextInput
Method: widgetTextInput Constructor
Params: Boolean editable - Flags if this is editable or not.
150

```

```

155 Boolean disabled
      Number tabIndex
      ValidationManager validationRule
      Boolean validateOnKeyPress
      String category
160
      Number size
165
      Number maxLength
      Anything value
      Return:
170 Description: Our text widget.
      *****
      function widgetTextInput(editable, disabled, tabIndex, validationRule,
                               validateOnKeyPress, category, size, maxLength, value)
      {
175 // Set the defaults for parameters that are not required.
      this.editable = true;
      this.disabled = false;
      this.validationRule = null;
      this.validateOnKeyPress = false;
180 this.tabIndex = null;
      this.focusText = null; // for IE bug - see OnBlur & OnChange

      // Set the values that have been defined.
185 if (editable != null)
      {
          this.editable = editable;
      }
190 if (disabled != null)
      {
          this.disabled = disabled;
      }
195 if (validationRule)
      {
          this.validationRule = validationRule;
      }
200 if (validateOnKeyPress)

```

```

        this.validateOnKeyPress = validateOnKeyPress;
    }

    // Other fields.
    this.listenerObj = null;

    // Create the element.
    this.element = document.createElement("INPUT");

    // Set some attributes.
    this.element.type = "text";

    if (maxLength)
    {
        this.element.maxLength = maxLength;
    }

    if (tabIndex != null)
    {
        this.tabIndex = this.element.tabIndex = tabIndex;
    }

    this.element.cjowner = this;

    CJElementHelper.registerElement(this.element, "cjowner");

    // Event handlers.
    this.element.onChange = _widgetTextInput_InputElement_OnChange;
    this.element.onblur = _widgetTextInput_InputElement_OnBlur;
    this.element.onfocus = _widgetTextInput_InputElement_OnFocus;

    if (this.validateOnKeyPress == true)
    {
        this.element.onkeypress = _widgetTextInput_InputElement_OnKeyPress;
    }

    // Styles.
    // If category is not defined then use the default.
    if (!category)
    {
        category = widgetTextInput.CATEGORY;
    }

    this.category = category;

    CJSetClass(this.element, this.category + widgetTextInput.STYLE_EXT_BASE, true);

    // Handle editable, disabled and so on.....
    this.setEditable(this.editable, true);
    this.setDisabled(this.disabled, true);

```

```

255     this.element.value = "";
        if (value != null)
        {
            this.element.value = value;
        }

        this.setSize(size);
        // Check for validation.
        this.validateValue();
    }

265
    /*****
    Constants
    *****/
270
    // Category used if one is not specified by the user.
    WidgetTextInput.CATEGORY = "WidgetTextInput";
275
    /*****
    Basic style extension.
    "-writable" = Writable style for the control (Main container).
    "-invalid" = Invalid style for the control (Main container).
    "-focus" = Focus style for the control (Main container).
    "-disabled" = Disabled style for the control (Main container).
    *****/
285

    // Extensions for the styles.
    WidgetTextInput.STYLE_EXT_BASE = "-base";
    WidgetTextInput.STYLE_EXT_WRITABLE = "-writable";
    WidgetTextInput.STYLE_EXT_INVALID = "-invalid";
    WidgetTextInput.STYLE_EXT_FOCUS = "-focus";
    WidgetTextInput.STYLE_EXT_DISABLED = "-disabled";
295

    /*****
    Public Methods definitions
    *****/
300

```



```

305 /*****
    Class : widgetTextInput
    Method: applyOnBlurStyle
    Params:
    Return:
    Description: Applies the on blur style to the element.
    *****/

310 function _widgetTextInput_ApplyOnBlurStyle()
{
    CJSetClass(this.element, this.category + widgetTextInput.STYLE_EXT_FOCUS, false);
315 }

320 /*****
    Class : widgetTextInput
    Method: applyOnFocusStyle
    Params:
    Return:
    Description: Applies the on focus style to the element.
    *****/

325 function _widgetTextInput_ApplyOnFocusStyle()
{
    CJSetClass(this.element, this.category + widgetTextInput.STYLE_EXT_FOCUS, true);
330 }

335 /*****
    Class : widgetTextInput
    Method: getDisabled
    Params:
    Return: Boolean true or false.
    Description: Returns the disabled attribute.
    *****/

340 function _widgetTextInput_GetDisabled()
{
    return (this.disabled);
345 }
350

```

```

355 /*****
    Class : widgetTextInput
    Method: getEditable
    Params:
    Return: Boolean true or false.
    Description: Returns the editable attribute.
    *****/
360

```

```

    function _widgetTextInput_GetEditable()
    {
365         return (this.editable);
    }

```

```

370 /*****
    Class : widgetTextInput
    Method: getValue
    Params:
    Return: Any value;
    Description: Returns this.element.value.
    *****/
375

```

```

380 function _widgetTextInput_GetValue()
    {
        return (this.element.value);
    }

```

385

```

/*****
    Class : widgetTextInput
    Method: invalidateValue
    Params:
    Return:
    Description: Sets the invalid value flag. It also sets invalid value style.
    *****/
390

```

395

```

    function _widgetTextInput_InvalidateValue()
    {
        if (this.valueIsValid == true)
395         {
400

```

```

405     this.valueIsValid = false;
        CJSetClass(this.element, this.category + widgetTextInput.STYLE_EXT_INVALID, true);
    }
}

410 /*****
    Class : widgetTextInput
    Method: isInputValid
    Params:
    Return: Boolean true or false.
    Description: Returns the validation value that is current known.
415 *****/

function _widgetTextInput_IsValueValid()
420 {
    return (this.valueIsValid);
}

425 /*****
    Class : widgetTextInput
    Method: setDisabled
    Params: Boolean disabled - true or false.
           Boolean force - true or false whether to always set the state
                        whether the state is changing or not.
    Return:
    Description: Sets the disabled attribute.
435 *****/

function _widgetTextInput_SetDisabled(disabled, force)
440 {
    // Only do something if it is changing.
    if ((this.disabled != disabled) || (force == true))
    {
        this.disabled = disabled;
    }
    CJSetClass(this.element, this.category + widgetTextInput.STYLE_EXT_DISABLED, this.disabled);
445
    // Check the tabbing index to make sure that the user will not be
    // able to tab to it if it is not editable.
    if (disabled == true)
    {
450

```

```

        this.element.tabIndex = -1;
    }
    else
    {
        if ((this.tabIndex != null) && (this.editable == true))
        {
            this.element.tabIndex = this.tabIndex;
        }
    }
}

455
460 }

465 /*****
Class : WidgetTextInput
Method: setEditable
Params: Boolean editable - true or false.
        Boolean force     - true or false whether to always set the state
                           - whether the state is changing or not.
Return:
Description: Sets the editable attribute.
*****/

470
475

function _widgetTextInput_setEditable(editable, force)
{
    // Only do something if it is changing.
    if ((this.editable != editable) || (force == true))
    {
        this.editable = editable;
        this.element.readOnly = !this.editable;
        CJSClass(this.element, this.category + WidgetTextInput.STYLE_EXT_WRITABLE, this.editable);
    }
    // Check the tabbing index to make sure that the user will not be
    // able to tab to it if it is not editable.
    if (editable == false)
    {
        this.element.tabIndex = -1;
    }
    else
    {
        if ((this.tabIndex != null) && (this.disabled == false))
        {
            this.element.tabIndex = this.tabIndex;
        }
    }
}

500 }

```

```

    }

505 /*****
    Class : widgetTextInput
    Method: setListener
    Params: Function listenerObj - Listener object.
    Return:
510 Description: Sets the listener object for event callbacks.
    The listener object should have any of the following
    methods defined:

515      onChangeTextChange
      onChangeTextBlur
      onChangeTextFocus
    *****/

520 function _widgetTextInput_SetListener(listenerObj)
{
    this.listenerObj = listenerObj;
525 }

/*****
    Class : widgetTextInput
    Method: setSize
    Params: Number size          - size to set the input field to in characters.
    Return:
    Description: Sets the size of the input field. If it is 0 then it will set it
                 to fit the current value of the field.
535 *****/

function _widgetTextInput_SetSize(size)
540 {
    if (size == null)    // default is auto size
    {
        size = 0;
    }

545     this.size = size;

    if (this.size)
    {
550         //this.element.size = this.size;
    }
}

```

```

        this.element.style.width = ((this.size +1)*.6)+"em";
    }
    else
    {
        if (this.size == 0)
        {
            if (this.element.value != null)
            {
                //this.element.size = this.element.value.toString().length + 1;
                this.element.style.width = ((this.element.value.toString().length+1) *.6) + "em";
            }
        }
    }
}
555
560
565

/*****
Class : WidgetTextInput
Method: setStyleState
Params: String styleClass - Class of the style.
        Boolean state     - state to set it to.
Return:
Description: Sets the style state. The style must exist.
*****/
570
575

function _widgetTextInput_SetStyleState(styleClass, state)
580 {
    CJSetClass(this.element, styleClass, state);
}

585
/*****
Class : WidgetTextInput
Method: setValue
Params: Any value.
Return: Any value;
Description: Sets the value of the field.
*****/
590

function _widgetTextInput_SetValue(value)
595 {
    if (value == null)
    {
        value = "";
    }
    600

```

```

    }
    this.element.value = value;

    // Check validation.
    this.validateValue();

    // Check if the size of the element has to be adjusted.
    if (this.size == 0)
    {
        this.setSize(this.size);
    }
}

/*****
Class : WidgetTextInput
Method: toElement
Params:
Return: HTMLInputElement
Description: Returns the main element associated with this widget.
*****/

605
610
615
620
625
630 }

function _widgetTextInput_ToElement()
{
    return (this.element);
}

/*****
Class : WidgetTextInput
Method: focus
Params: none
Return: none
Description: Gives the text input element the focus
*****/
635
640 function _widgetTextInput_Focus()
{
    this.element.focus();
}

/*****
Private Methods definitions
*****/
645
650

```

```

/*****
Class : widgetTextInput
Method: onChange
Params:
Return:
Description: This function is called on the onChange event. It checks
for validations.
*****/
655
660

```

```

function _widgetTextInput_OnChange(e)
{
    // no need to test for change on blur
    this.focusText = null;

    // Check the validation rule for the text.
    this.validateValue();

    // Resize the field if necessary.
    this.setSize(this.size);

    // Check if there is a callback set for this event.
    if (this.listenerObj != null)
    {
        // Check if the method is defined.
        if (this.listenerObj.onTextInputChange)
        {
            this.listenerObj.onTextInputChange(this);
        }
    }
}
685

```

```

/*****
Class : widgetTextInput
Method: onBlur
Params:
Return:
Description: This function is called when the text box loses focus.
It also checks for validations.
*****/
690
695

```

```

function _widgetTextInput_OnBlur(e)
700 {

```



```

705 // Test to see if value changed but no onChange call was made (IE bug)
    if ( this.focusText != null )
    {
        if ( this.getValue() != this.focusText )
        {
            this.onChange( e );
        }
        this.focusText = null;
    }
    this.applyOnBlurStyle();
710
    // Check if there is a callback set for this event.
    if (this.listenerObj != null)
    {
        // Check if the method is defined.
        if (this.listenerObj.onTextInputBlur)
        {
            this.listenerObj.onTextInputBlur(this);
720        }
    }

    // This is needed because if the value of the element is changed during
    // the onChangeCallback event, the master value of the input box
    // is not reset.
    this.element.value = this.element.value;
725
}

730 /*****
    Class : WidgetTextInput
    Method: onFocus
    Params:
    Return:
    Description: This function is called when the text box gets focus.
                It also checks for validations.
    *****/
740
function _widgetTextInput_OnFocus(e)
{
    var cJEvent = new CJEvent(e);
745
    if ((this.editable == true) && (this.disabled == false))
    {
        this.applyOnFocusStyle();
750
        // pull out the current text to save for blur test.

```

```

755 // IE will not call onChange if you click outside the IFRAME!!
    this.focusText = this.getValue();

    // Check if there is a callback set for this event.
    if (this.listenerObj != null)
    {
        // Check if the method is defined.
        if (this.listenerObj.onTextInputFocus)
        {
            this.listenerObj.onTextInputFocus(this);
        }
    }
    else
    {
        if (this.disabled == true)
        {
            this.element.blur();
        }
    }
}

775 /*****
Class : WidgetTextInput
Method: onKeyPress
Params:
780 Return: This function is called when a key is pressed.
Description: It also checks for validations.
*****/

785 function _WidgetTextInput_OnKeyPress(e)
{
    var cJEvent = new CJEvent(e);
    return (this.validateKeyPress(cJEvent));
}

795 /*****
Class : WidgetTextInput
Method: validateKeyPress
Params:
800 Return:

```

```

Description: Checks if the key pressed passes the validation rule. If it
            fails the validation, the key press will not be allowed.
            *****/
805
function _widgetTextInput_ValidateKeyPress(cjEvent)
{
    var keyChar;
810    // If not the return key then process it.
    if (cjEvent.keyCode != 13)
    {
        keyChar = String.fromCharCode(cjEvent.keyCode);
815        if (this.validationRule != null)
        {
            if (this.validationRule.isInvalidChar(keyChar) == false)
            {
                return (false);
820            }
            else
            {
                cjEvent.preventDefault();
825            }
        }
        return (true);
    }
830
    /*****
    Class : WidgetTextInput
    Method: validateValue
    Params:
    Return:
    Description: Checks if the value currently in the text box passes the
                validation rule.
    *****/
835
function _widgetTextInput_ValidateValue()
{
    this.valueIsValid = true;
840    if (this.validationRule != null)
    {
        if (this.validationRule.isInvalidString(this.element.value) == false)
850

```

```

        this.valueIsValid = false;
    }
}

855  CJsetClass(this.element, this.category + WidgetTextInput.STYLE_EXT_INVALID, !this.valueIsValid);
    }

860  /*****
    Event handlers
    *****/
865  /*****
    Class : WidgetTextInput
    Method: InputElement_OnXXX
    Params:
    Return: Child result.
    Description: Static private methods use to invoke the object methods
    on the input element.
    *****/
870  /*****
    *****/
875  /*****

function _widgetTextInput_InputElement_OnChange(e)
{
    return (this.cjowner.onChange(e));
}

function _widgetTextInput_InputElement_OnBlur(e)
{
    return (this.cjowner.onBlur(e));
}

function _widgetTextInput_InputElement_OnFocus(e)
{
    return (this.cjowner.onFocus(e));
}

function _widgetTextInput_InputElement_OnKeyPress(e)
{
    return (this.cjowner.onKeyPress(e));
}

900  /*****

```

```

Public Methods
*****

905 widgetTextInput.prototype.applyOnBlurStyle = _widgetTextInput_ApplyOnBlurStyle;
    widgetTextInput.prototype.applyOnFocusStyle = _widgetTextInput_ApplyOnFocusStyle;
    widgetTextInput.prototype.getDisabled = _widgetTextInput_GetDisabled;
    widgetTextInput.prototype.setEditable = _widgetTextInput_GetEditable;
910 widgetTextInput.prototype.getValue = _widgetTextInput_GetValue;
    widgetTextInput.prototype.invalidateValue = _widgetTextInput_InvalidateValue;
    widgetTextInput.prototype.isValueValid = _widgetTextInput_IsValueValid;
    widgetTextInput.prototype.setDisabled = _widgetTextInput_SetDisabled;
    widgetTextInput.prototype.setEditable = _widgetTextInput_SetEditable;
915 widgetTextInput.prototype.setListener = _widgetTextInput_SetListener;
    widgetTextInput.prototype.setSize = _widgetTextInput_SetSize;
    widgetTextInput.prototype.setStyleState = _widgetTextInput_SetStyleState;
    widgetTextInput.prototype.setValue = _widgetTextInput_SetValue;
    widgetTextInput.prototype.toElement = _widgetTextInput_ToElement;
920 widgetTextInput.prototype.focus = _widgetTextInput_Focus;

/*****
Private Methods
*****

925 widgetTextInput.prototype.onChange = _widgetTextInput_OnChange;
    widgetTextInput.prototype.onBlur = _widgetTextInput_OnBlur;
    widgetTextInput.prototype.onFocus = _widgetTextInput_OnFocus;
    widgetTextInput.prototype.onKeyPress = _widgetTextInput_OnKeyPress;
    widgetTextInput.prototype.validateValue = _widgetTextInput_ValidateValue;
    widgetTextInput.prototype.validateKeyPress = _widgetTextInput_ValidateKeyPress;
935

/*****
Public Methods definitions
*****

940

/***** END widgetTextInput Class *****/

945

/*****
Class : widgetTextDisplay
*****

950

```

```

*****/
/*****
955 Class : WidgetTextDisplay
Method: WidgetTextDisplay Constructor
Params: String category - This is a string identifying
Anything value - Initial value of the field.
960 Return: Our text display widget.
Description: *****/

function WidgetTextDisplay(category, value)
{
    // Create the container element.
    this.element = document.createElement("SPAN");
    // Set some attributes.
    this.element.type = "text";
    // Create the text element.
    this.textElement = document.createTextNode(value);
    // Styles.
    // If category is not defined then use the default.
    if (category)
    {
        category = WidgetTextDisplay.CATEGORY;
    }
    this.category = category;
    cJsetClass(this.element, this.category + WidgetTextDisplay.STYLE_EXT_BASE, true);
    // Stick the text node into the span.
    this.element.appendChild(this.textElement);
990 }

/*****
995 Constants
*****/

1000 // Category used if one is not specified by the user.

```

```

1005 widgetTextDisplay.CATEGORY = "widgetTextDisplay";

1010 /*****
      Basic style names.
      ".base" = Base style for the control (Main container).
      *****/

1015 // Extensions for the styles.
      widgetTextDisplay.STYLE_EXT_BASE = "-base";

1020 /*****
      Public Methods definitions
      *****/

1025 /*****
      Class : widgetTextDisplay
      Method: getValue
      Params:
      Return: Any value;
      Description: Returns this.element.value.
      *****/

1030 function _widgetTextDisplay_GetValue()
{
1035     return (this.textElement.data);
}

1040 /*****
      Class : widgetTextDisplay
      Method: setDisabled
      Params:
      Return: Any value;
      Description:
      *****/

1045 function _widgetTextDisplay_SetDisabled()

```

```

{
    // Does absolutely nothing right now.
}

1055
/*****
Class : WidgetTextDisplay
Method: setStyleState
Params: String styleClass - Class of the style.
        Boolean state     - state to set it to.
Return:
Description: Sets the style state. The style must exist.
*****/
1065

function _WidgetTextDisplay_setStyleState(styleClass, state)
{
    CJSClass(this.element, styleClass, state);

1070 }

1075 /*****
Class : WidgetTextDisplay
Method: setValue
Params: Any value.
Return: Any value;
Description: Sets the value of the field.
*****/
1080

1085 function _WidgetTextDisplay_setValue(value)
{
    if (value == null)
    {
        value = "";
    }

    this.textElement.data = value;
}

1095

/*****
Class : WidgetTextDisplay
Method: toElement
Params:
1100

```



```

Return: HTML element
Description: Returns the main element associated with this widget.
*****
1105
function _widgetTextDisplay_ToElement()
{
    return (this.element);
1110 }

/***** Private Methods definitions *****/
1115 *****/

1120 /***** Public Methods prototypes *****/
Public Methods prototypes
*****/

1125 widgetTextDisplay.prototype.getValue = _widgetTextDisplay_GetValue;
widgetTextDisplay.prototype.setDisabled = _widgetTextDisplay_SetDisabled;
widgetTextDisplay.prototype.setStyleState = _widgetTextDisplay_SetStyleState;
widgetTextDisplay.prototype.setValue = _widgetTextDisplay_SetValue;
1130 widgetTextDisplay.prototype.toElement = _widgetTextDisplay_ToElement;

/***** Private Methods prototypes *****/
1135 *****/

1140 /***** END widgetTextDisplay Class *****/

```

Exhibit 10

1 /***** AUTOMATED LOGIC CORPORATION *****/

5 Copyright (c) 1999 - 2000 All Rights Reserved
This document contains confidential/proprietary information. *****/

@(#)ControlTextInput.js
Initial Author: Jide Omisore

10 Description: Text Input Box control.

Dependencies:

15 /util/browserapi.js
/util/cjelementhelper.js
/util/compat.js
/controls/CJEvent.js
/controls/exception.js
/controls/iconhelper.js
/controls/stylemgr.js
20 /controls/widgettextinput.js

\$Log: /green/webroot/common/controls/ControlTextInput.js \$

25 * 29 11/29/99 3:14p Ddavidson
* added focus() methods to focus to (first) text field
*
* 28 11/18/99 9:05p Jomisore
* Fixed it so that 0 shows a plus sign when the force plus sign option is
30 * set to true.
*
* 27 11/18/99 10:15a Tirish
*
* 26 10/05/99 10:14a Jomisore
35 * Moved multiplystring to BrowserApi.js
*
* 25 9/24/99 10:48a Jomisore
* The controls now take a category in the constructor.
*
40 * 24 9/22/99 3:40p Jomisore
* Made changes to the style stuff so that things run faster.
*
* 23 9/22/99 12:01p Sappling
* This has been changed to partially implement the new styles
45 * 22 9/17/99 3:24p wdoover

```

*      21      9/10/99 4:47p wdoover
*      *      Modified to handle bad values
50      *
*      20      9/09/99 9:49a Jomisore
*      *      Cleaned up code for the code review. Added the octet string control.
*      *      Added restrictions to the controls.
*
*      19      9/01/99 3:57p Jomisore
55      *      Changed it so that the control's will display "?" and will be disabled
*      *      if primname is null.
*
*      18      8/23/99 2:36p wdoover
60      *      Adding ControlBitstring class
*
*      17      8/16/99 2:09p Jomisore
*      *      Changed getValueText() to toString() for the primitives.
*
65      *      16      8/13/99 11:11a Jomisore
*      *      Fixed a problem with formatting when 0 is the value if the number edit
*      *      control.
*
*      15      8/11/99 9:20a Jomisore
70      *      Added the "undefined" style for textdisplay widget.
*
*      13      8/09/99 5:41p Jomisore
*      *      Removed the set and get editable methods.
*
75      *      12      8/09/99 9:17a Jomisore
*      *      Added a TextDisplaywidget.
*
*      11      8/06/99 11:43a Jomisore
*      *      Made the number formatting simpler.
80      *
*      10      8/05/99 6:01p Jomisore
*      *      Changed it so that the digits specified that go to the left and right
*      *      of the decimal are formatting options and not restrictions. Although
*      *      when it is truncated the number is not rounded up (this will be worked
85      *      on).
*
*      9      8/05/99 10:19a Jomisore
*      *      Updated a comment.
*
90      *      8      8/04/99 5:49p Jomisore
*      *      Updated style management.
*

```

```

7      8/02/99 2:02p Jomisoire
* Finished up on the number edit box.
*
95  *
* 6      7/29/99 5:28p Jomisoire
* Made ControlTextInput a base class and added ControlCharStrEdit and
* ControlNumberEdit to inherit from the base class.
*
100 * 5      7/29/99 10:09a Jomisoire
* Changed "affectstyle" to "changelstyle".
*
* 4      7/28/99 10:46a Jomisoire
* Added the affectstyleById method.
105 *
* 3      7/28/99 9:18a Jomisoire
* Added some methods.
*
110 * 2      7/23/99 6:30p Jomisoire
* Added comments.
*
* 1      7/23/99 6:26p Jomisoire
* Text Input Control.
* 1      07/23/99 10:55p Jomisoire
115 * Initial implementation
*
*****/

120 /*****
Class : ControlTextInput

125 Public methods:
onTextInputChange
setDisabled
toElement
valueChanged

130 *****/

/*****
Class : ControlTextInput
135 Method: ControlTextInput - Constructor
Params: Object prim - Primitive object.
Boolean editable - Flags if this is editable
or not.
*****/

```

```

140 Boolean disabled - Flags if this is disabled
                        or not.
Number tabIndex - Tab index.
String category - Category used by the text input widget
                  if one is created.
String textSpanCategory - Category used by the text display widget
                          if one is created.
Number size - Size of the text field in characters.
Number maxLength - Maximum number of characters that the user
                  can input.
ValidationManager validationMgr - Validation manager.
Boolean validateOnKeyPress - Validate each key stroke?

150 Return:
Description: Base class for the text input controls. The following controls
            should inherit from it:

155 ControlCharStrEdit
ControlNumberEdit
ControlBitStrEdit
ControlOctetStrEdit

160 *****
function ControlTextInput(prim, editable, disabled, tabIndex, category,
                        size, maxLength, validationMgr,
                        validateOnKeyPress)
{
165     var message;
    var undefined;

    this.ControlBase(editable, disabled, tabIndex);
    // Required fields.
    // Find the primitive.
    this.prim = prim;

    if (!this.prim)
    {
175         // If this parameter is undefined then this is being called when the
        // prototype object for the derived controls are being initialized.
        if (prim === undefined)
        {
            return;
        }
        else
        {
180             size = 0;
            this.disabled = true;

```

```

185     }
    } else
    {
        // Add this control to the primitive's list of listeners.
        this.prim.addListener(this);
190
        if (this.prim.iswritable() == false)
        {
            // If the primitive is not writable then force the editable to be false.
            this.editable = false;
195        }
    }
}

200 this.category = category;

    if ((editable == true) || (editable == null))
    {
        this.textwidgetCategory = "-" + widgetTextInput.CATEGORY;
205        // Add the text widget.
        this.widgets[0] = new WidgetTextInput(this.editable, this.disabled, this.tabIndex,
            validationMgr, validateOnKeyPress,
            this.category + this.textwidgetCategory,
            size, maxLength);
210
        this.oldTabIndex = this.tabIndex;
        this.widgets[0].setListener(this);
    }
    else
    {
215        this.textwidgetCategory = "-" + widgetTextDisplay.CATEGORY;

        // Use a text display instead of a text input.
        this.widgets[0] = new WidgetTextDisplay(this.category + this.textwidgetCategory);
220    }

    // Fill in the value of the primitive if there is one.
    this.valueChanged();
225
}

/*****
Constants
*****/
230 *****/

```

```

/*****
Basic style extension.
235 "-writable" = Writable style for the control's widget.
    "-invalid" = Invalid style for the control's widget.
    "-focus" = Focus style for the control's widget.
    "-disabled" = Disabled style for the control's widget.
    "-undefined" = Undefined style for the control's widget.
    "-badvalue" = Badvalue style for the control's widget.
*****/

240 *****/

245 // Extensions for the styles.
    ControlTextInput.STYLE_EXT_BASE = "-base";
    ControlTextInput.STYLE_EXT_WRITABLE = "-writable";
    ControlTextInput.STYLE_EXT_INVALID = "-invalid";
    ControlTextInput.STYLE_EXT_FOCUS = "-focus";
    ControlTextInput.STYLE_EXT_DISABLED = "-disabled";
    ControlTextInput.STYLE_EXT_UNDEFINED = "-undefined";
    ControlTextInput.STYLE_EXT_BADVALUE = "-badvalue";

255 /*****
Public Methods definitions
*****/

260 /*****
Class : ControlTextInput
Method: onTextInputChange
Params:
Return:
Description: This is called by the text widget when the value of the text
            input box changes.
*****/

270

function _ControlTextInput_OnTextInputChange()
{
    var value;

```



```

CJAssert(this.prim, "There has to be a primitive defined.");
    if (this.widgets[0].isValueValid())
    {
        value = this.widgets[0].getValue();
        this.valid = true;
        try
        {
            this.prim.setValue(value, this);
            // Clear the undefined state.
            if (value != null)
            {
                this.widgets[0].setState(this.category + this.textWidgetCategory
                    + ControlTextInput.STYLE_EXT_UNDEFINED,
                    false);
            }
        }
        catch (ex)
        {
            if (ex instanceof CJException.InvalidValue)
            {
                this.widgets[0].invalidateValue();
                this.valid = false;
            }
            else
            {
                throw (ex);
            }
        }
    }
    else
    {
        this.valid = false;
    }
}

315 /*****
    Class : ControlTextInput
    Method: setDisabled
    Params:
    Return:
    Description: Gets the disabled attribute.
    *****/

```

```

325 function _ControlTextInput_SetDisabled(disabled)
    {
        CAssert((disabled == false) || (disabled == true),
            "disabled must be true or false");
330         this.disabled = disabled;

        this.widgets[0].setDisabled(this.disabled);
335     }

/*****
340     Class : ControlTextInput
        Method: toElement
        Params:
        Return:
        Description: Returns the main element.
345     *****/

function _ControlTextInput_ToElement()
350 {
    return (this.widgets[0].toElement());
}

355 /*****
        Class : ControlTextInput
        Method: focus
        Params:
        Return:
        Description: Give the control the focus
360     *****/

365 function _ControlTextInput_Focus()
    {
        return (this.widgets[0].focus());
    }

```

```

370 /*****
    Class : ControlTextInput
    Method: valueChanged
    Params:
    Return:
    Description: This is a callback that is registered with the associated
                 primitive that is called when the value of the primitive changes.
    *****/
380

function _ControlTextInput_valueChanged()
{
    385     var value;

    if (this.prim)
    {
        390         if (this.prim.hasBadValue())
        {
            this.widgets[0].setStyleState(this.category + this.textWidgetCategory
                                           + ControlTextInput.STYLE_EXT_BADVALUE, false);

            395             this.badValue = true;
        }
        else if (this.badValue)
        {
            400             this.widgets[0].setStyleState(this.category + this.textWidgetCategory
                                                   + ControlTextInput.STYLE_EXT_BADVALUE, false);

            this.badValue = false;
        }
        // If the value is undefined then the value returned will be "".
        value = this.getPrimitiveText(this.prim);
        405         if (this.checkUndefined() == false)
        {
            this.widgets[0].setValue(value);
        }
        else
        {
            410             this.widgets[0].setValue("?");
        }
    }
}

```

```

415     }
    else
    {
        this.widgets[0].setValue("?");
    }
420 }

425 /*****
Private Methods definitions
*****/

430 /*****
Class : ControlTextInput
Method: checkUndefined
Params:
Return: Boolean true or false.
Description: Checks if the primitive is undefined. If it is, it sets the
            undefined adapter state to true.
*****/

440 function _ControlTextInput_CheckUndefined()
{
    var undefined;
    CAssert(this.prim, "There has to be a primitive defined.");
    undefined = this.prim.isundefined();
450     this.widgets[0].setState(this.category + this.textWidgetCategory +
        ControlTextInput.STYLE_EXT_UNDEFINED, undefined);
    return (undefined);
455 }

/*****
Class : ControlTextInput
Method: getPrimitiveText
Params:
460

```

```

465 Return: primitive value.
    Description: Get's the text value of the prim. Override this to so any special
    formatting with the primitive value.
    *****/
function _ControlTextInput_getPrimitiveText(prim)
{
    470     var value;

    CJAssert(prim, "There has to be a primitive defined.");

    475     value = prim.toString();

    return (value);
}

480 /*****
    Public Methods prototypes
    *****/
ControlTextInput.prototype = new ControlBase();
ControlTextInput.prototype.ControlTextInput = ControlTextInput;

490 ControlTextInput.prototype.onTextInputChange = _ControlTextInput_onTextInputChange;
ControlTextInput.prototype.setDisabled = _ControlTextInput_SetDisabled;
ControlTextInput.prototype.toElement = _ControlTextInput_ToElement;
ControlTextInput.prototype.focus = _ControlTextInput_Focus;
ControlTextInput.prototype.valueChanged = _ControlTextInput_valueChanged;

495
    *****/
Private Methods prototypes
    *****/

500 ControlTextInput.prototype.checkUndefined = _ControlTextInput_CheckUndefined;
    ControlTextInput.prototype.getPrimitiveText = _ControlTextInput_getPrimitiveText;

```

```

510 /***** END ControlTextInput Class *****/
515 /*****
Class : ControlCharStrEdit
Method: ControlCharStrEdit - Constructor
Params: String primname
Boolean editable
Boolean disabled
Number tabIndex
Number size
Return:
Description: Character String edit Control.
*****/
525 /*****
Class : ControlCharStrEdit
Method: ControlCharStrEdit - Constructor
Params: String primname
Boolean editable
Boolean disabled
Number tabIndex
Number size
Return:
Description: Character String edit Control.
*****/
530 function ControlCharStrEdit(primname, editable, disabled, tabIndex, size, category)
{
var prim;
var maxLength;
535 if (category == null)
{
category = ControlCharStrEdit.CATEGORY;
}
540 if (primname == null)
{
prim = null;
}
else
{
545 // Get the prim.
prim = PrimitiveBase.findPrimitive(primname);
// Make sure we have the correct type for the primitive.
if (prim instanceof PrimitiveString)
{
550 // This is ok.
}
}
}

```

```

else
{
555 // Bad primitive.
    message = "Primitive \" + primname + "\" type is invalid for Number Edit controls";
    throw new CJException.InvalidValue(message);
}
560 // Get the maximum number of characters that can be entered.
    if (prim.hasMaxLength())
    {
565         maxLength = prim.getMaxLength();
    }
    // Initialize the base class.
    this.ControlTextInput(prim, editable, disabled, tabIndex,
570         category, size, maxLength);
}

575 // Inherit from text input control.
ControlCharStEdit.prototype = new ControlTextInput();

580 /*****
    Constants
    *****/

585 // Styles.
ControlCharStEdit.CATEGORY = "ControlCharStr";

590 /***** END ControlCharStEdit Class *****/

595 /*****
    Class : ControlNumberEdit
    Method: ControlNumberEdit - Constructor
    Params: String primname
            - Name of the primitive.
    *****/

```

```

600         Boolean editable
        Boolean disabled
        Number tabIndex
        Number size
        Number digLeftOfDecimal
        Number digRightOfDecimal
        Boolean showPlusSign

        - Flags if this is editable
        - or not.
        - Flags if this is disabled
        - or not.
        - Tab index.
        - Size of the text field in characters.
        - Minimum number of digits to the
        - left of the decimal point.
        - The number of digits that have
        - to be to the right of the
        - decimal point.
        - Boolean indicating whether to
        - automatically show a plus sign
        - or not.

        Return:
        Description: Number edit Control.
615 *****/*****
function ControlNumberEdit(primname, editable, disabled, tabIndex, size,
        digLeftOfDecimal, digRightOfDecimal, showPlusSign, category)
{
    var prim;
    var validationRule = null;
    var validationMgr = null;
    var primitiveMax;
    var primitiveMin;
    var primitivePrecision = null;

    if (category == null)
    {
        category = ControlNumberEdit.CATEGORY;
    }

    if (primname == null)
    {
        prim = null;
        digLeftOfDecimal = 0;
        digRightOfDecimal = null;
        showPlusSign = false;
    }
    else
    {
        // Get the prim.

```



```

645 prim = PrimitiveBase.findPrimitive(primname);
    // Make sure we have the correct type for the primitive.
    if (prim instanceof PrimitiveNumber)
    {
        // This is ok.
    }
    else
    {
        // Bad primitive.
        message = "Primitive \"\" + primname + "\" type is invalid for Number Edit controls";
        throw new CJException.InvalidValue(message);
    }

660 // Get restrictions.
    primitiveMin = prim.getMinimum();
    primitiveMax = prim.getMaximum();

665 if (prim.hasPrecision() == true)
    {
        primitivePrecision = prim.getPrecision();

        validationRule = new RuleRange(primitiveMin, primitiveMax);
        validationMgr = new ValidationManager([validationRule]);
    }

675 // Formatting options.
    if (!digLeftofDecimal)
    {
        digLeftofDecimal = 0;
    }

680 this.digLeftofDecimal = digLeftofDecimal;
    this.digRightofDecimal = digRightofDecimal;

685 if (primitivePrecision != null)
    {
        this.digRightofDecimal = primitivePrecision;

        this.showPlussign = false;

690 if (showPlussign)

```

```

{
    this.showPlusSign = showPlusSign;
}

695 // Initialize the base class.
    this.ControlTextInput(prim, editable, disabled, tabIndex,
        category, size, null,
        validationMgr,
        true);
700 }

// Inherit from text input control.
705 ControlNumberEdit.prototype = new ControlTextInput();

/*****
710 Constants
*****/

715 // Styles.
    ControlNumberEdit.CATEGORY = "ControlNumberEdit";

720 /*****
    Public Methods definitions
*****/

725 /*****
    Class : ControlNumberEdit
    Method: getPrimValueText
    Params:
    Return:
    Description: Get's the text value of the prim. It also formats the value
        according to what th user wants.
*****/

735

```

```

function _ControlNumberEdit_getPrimitiveText(prim)
{
    var value;
    740 CJAssert(prim, "There has to be a primitive defined.");
        value = prim.toString();
    745 value = this.formatValue(value);
        return (value);
    }
    750

    /*****
    Class : ControlNumberEdit
    Method: onTextInputChange
    Params:
    Return:
    Description: This is called by the text widget when the value of the text
    input box changes.
    *****/
    755
    760

function _ControlNumberEdit_onTextInputChange()
{
    var stringValue;
    765 CJAssert(this.prim, "There has to be a primitive defined.");
        stringValue = this.widgets[0].getValue();
    770 // If the text entered is invalid then do not bother to set the primitive.
        if (this.widgets[0].isInvalid() == true)
        {
            stringValue = this.formatValue(stringValue);
    775 // Set the value of the text widget.
            this.widgets[0].setValue(stringValue);
            this.valid = true;
            try
            {
    780 // Set the value of the primitive.
                this.prim.setValue(stringValue, this);
            }
        }
    }
}

```

```

785 // Clear the undefined state.
    if (stringValue != null)
    {
        this.widgets[0].setState(this.category + this.textwidgetCategory
                                + ControlTextInput.STYLE_EXT_UNDEFINED, false);
    }
790 }
    catch (ex)
    {
        if (ex instanceof CJException.InvalidValue)
        {
795     this.widgets[0].invalidateValue();
            this.valid = false;
        }
        else
        {
800     throw (ex);
        }
    }
}
805 }
    else
    {
        this.valid = false;
    }
810 }

/***** Private Methods definitions *****/
815 *****/

820 /****
    Class : ControlNumberEdit
    Method: formatValue
    Params: String invalue - String to format.
    Return:
825 Description: Formats the value in the text field to however the user
                  specified it in the constructor. The input value has to be a
                  valid javascript number.
*****/

```

```

830     function _ControlNumberEdit_FormatValue(invalue)
831     {
832         var value = invalue;
833         var numberLiteral;
834         if ((value != null) && (value != ""))
835         {
836             // Put the decimal point in the right spot.
837             value = cjCompat.floatFix(value, this.digLeftOfDecimal, this.digRightOfDecimal);
838             // Add any plus signs.
839             // If the user wants to stick in the plus sign in front of positive
840             // numbers then do it.
841             if (this.showPlusSign == true)
842             {
843                 numberLiteral = new Number(value);
844                 if (numberLiteral >= 0.0)
845                 {
846                     // If plus sign isn't already there then stick it in.
847                     if (value.indexOf("+", 0) == -1)
848                     {
849                         value = "+" + value;
850                     }
851                 }
852             }
853             return (value.toString());
854         }
855         else
856         {
857             return (value);
858         }
859     }
860
861     /*****
862     Public Methods prototypes
863     *****/

```

```

875 *****/

ControlNumberEdit.prototype.getPrimValueText = _ControlNumberEdit_getPrimValueText;
880 ControlNumberEdit.prototype.onTextInputChange = _ControlNumberEdit_onTextInputChange;

/***** Private Methods prototypes *****/
885 Private Methods prototypes
*****/

890 ControlNumberEdit.prototype.formatValue = _ControlNumberEdit_formatValue;

/***** END ControlNumberEdit Class *****/
895 /*****
/***** Class : ControlBitStrEdit
This class provides a control for editing bit strings of variable
900 lengths.
PUBLIC METHODS:
onTextInputChange
905 *****/
/*****
Class : ControlBitStrEdit
Method: ControlBitStrEdit
Params: String primname
Boolean editable
Boolean disabled
Number tabIndex
Number size
Return:
915
920 *****/

```

```

function ControlBitStrEdit(primname, editable, disabled, tabIndex, size, category)
{
    var prim;
    var maxLength;
    var validationRule = null;
    var validationMgr = null;

    if (category == null)
    {
        category = ControlBitStrEdit.CATEGORY;
    }

    if (primname == null)
    {
        prim = null;
    }
    else
    {
        prim = PrimitiveBase.findPrimitive(primname);

        if ((prim instanceof PrimitiveBitString) == false)
        {
            message = "Primitive \"\"+primname+\"\" type is invalid for Bit String Edit Controls.";
            throw new CjException.InvalidValue(message);
        }

        validationRule = new RuleBase(/^[01]*$/ , /^[01]*$/);
        validationMgr = new ValidationManager([validationRule]);

        // Get the maximum number of characters that can be entered.
        maxLength = prim.getMaxBits();
    }

    this.ControlTextInput(prim, editable, disabled, tabIndex,
        category, size, maxLength,
        validationMgr, true);
}

// Inherit from text input control
ControlBitStrEdit.prototype = new ControlTextInput();

/*****

```

```

Constants
*****
970 // Styles.
    ControlBitStEdit.CATEGORY = "ControlBitStr";

975 /***** END ControlBitStEdit Class *****/

/*****
980 Class : ControlOctetStEdit

    This class provides a control for editing octet strings of variable
    lengths.

985 PUBLIC METHODS:
    onTextInputChange

*****
990 /*****
    Class : ControlOctetStEdit
    Method: ControlOctetStEdit - Constructor
    Params: String primname      - Name of octet string primitive.
            Boolean editable     - Flags if this is editable
                                or not.
                                - Flags if this is disabled
                                or not.
                                - Tab index.
                                - Size of the text field in characters.
                                - Whether to separate the bytes with spaces.
                                - Default is false.

                                Return:

1000

1005 *****
function ControlOctetStEdit(primname, editable, disabled, tabIndex, size,
                             separateBytes, category)
{
    var prim;
    var maxLength;
    var primMaxLength;
    var validationRule = null;

```



```

var validationMgr = null;

1015 if (category == null)
    {
        category = ControlOctetStrEdit.CATEGORY;
    }

1020 this.separateBytes = separateBytes;

    if (primname == null)
    {
        prim = null;
    }
    else
    {
        prim = PrimitiveBase.findPrimitive(primname);

1030 if ((prim instanceof PrimitiveOctetString) == false)
    {
        message = "Primitive \"\"+primname+\"\" type is invalid for Octet String Edit Controls.";
        throw new CjException.InvalidValue(message);
    }

1035 validationRule = new RuleOctet(2, true, true);
validationMgr = new ValidationManager([validationRule]);

    // Get the maximum number of characters that can be entered.
    // Max length will be maximum number of bytes * 2.
    primMaxLength = prim.getMaxLength();
    maxLength = primMaxLength * 2;

    // Factor in the spaces for separating the bytes..
    if (this.separateBytes == true)
    {
        maxLength += primMaxLength - 1;
    }

1050 }

this.ControlTextInput(prim, editable, disabled, tabIndex,
                      category, size, maxLength,
                      validationMgr, true);

1055 }
// Inherit from text input control

```

```

1060 ControlOctetStrEdit.prototype = new ControlTextInput();

1065 /**
    Constants
    */
    // styles..
    ControlOctetStrEdit.CATEGORY = "ControlOctetStr";

1070
    /**
    Public Methods definitions
    */
    /**
    Class : ControlOctetStrEdit
    Method: getPrimitiveText
    Params:
    Return:
    Description: Get's the text value of the prim. It also formats the value
    according to what th user wants.
    */
1080
1085
    function _ControlOctetStrEdit_getPrimitiveText(prim)
    {
        var value;

        CJAssert(prim, "There has to be a primitive defined.");

1095         value = prim.toString();
        value = this.formatValue(value);

1100         return (value);
    }

    /**

```

```

1105 Class : ControlOctetStrEdit
      Method: onTextInputChange
      Params:
      Return:
      Description: This is called by the text widget when the value of the text
                   input box changes.
1110 *****/

1115 function _ControlOctetStrEdit_OnTextInputChange()
    {
        var stringValue;

        // Get the value entered.
        stringValue = this.widgets[0].getValue();

        // Add or remove spaces.
        stringValue = this.formatValue(stringValue);

        // Set the value of the text widget with the formatted string.
        this.widgets[0].setValue(stringValue);

        // If the text entered is invalid then do not bother to set the primitive.
        if (this.widgets[0].isValueValid() == true)
        {
            // Remove spaces and stick it into the primitive.
            stringValue = stringValue.replace(new RegExp("[ ]", "g"), "");
            try
            {
                // Set the value of the primitive.
                this.prim.setValue(stringValue, this);

                // Clear the undefined state.
                if (stringValue != null)
                {
                    this.widgets[0].setState(this.category + this.textWidgetCategory
                                             + ControlTextInput.STYLE_EXT_UNDEFINED, false);
                }
            }
            catch (ex)
            {
                if (ex instanceof CJException.InvalidValue)
                {
                    this.widgets[0].invalidateValue();
                }
            }
        }
    }

```

```

1155         this.valid = false;
            }
            else
            {
                throw (ex);
            }
        }
    }
    else
    {
        this.valid = false;
    }
}
1165 }

/*****
1170 Private Methods definitions
*****/

1175 /*****
Class : ControlOctetStrEdit
Method: FormatValue
Params: String inval - String to format.
Return:
Description: Formats the value in the text field to however the user
              specified it in the constructor. The input value has to be a
              valid hex string.
*****/

1185 function _ControlOctetStrEdit_FormatValue(stringValue)
{
    var newValue = "";
    var chr;
    var nibbleCount;

    CJAssert(this.prim, "There has to be a primitive defined.");

    // Number of nibbles found.
1195

```

```

nibbleCount = 0;

// Format now.
for (var index = 0; index < stringValue.length; index++)
{
    chr = stringValue.charAt(index);

    if (chr != " ")
    {
        newValue += chr;
        ++nibbleCount;

        // This is a byte.
        if (nibbleCount == 2)
        {
            // If user wants to separate the bytes then add a space.
            if (this.separateBytes == true)
            {
                newValue += " ";
            }

            nibbleCount = 0;
        }
    }

    return (newValue);
}

1200 nibbleCount = 0;
1205 // Format now.
1210 for (var index = 0; index < stringValue.length; index++)
1215 {
1220     chr = stringValue.charAt(index);
1225     if (chr != " ")
1230     {
1235         newValue += chr;
1240         ++nibbleCount;
1245         // This is a byte.
1250         if (nibbleCount == 2)
1255         {
1260             // If user wants to separate the bytes then add a space.
1265             if (this.separateBytes == true)
1270             {
1275                 newValue += " ";
1280             }
1285             nibbleCount = 0;
1290         }
1295     }
1300     return (newValue);
1305 }

1310 // *****
1315 // Public Methods prototypes
1320 // ***** /

1325 ControlOctetStrEdit.prototype.getPrimValueText = _ControlOctetStrEdit_getPrimValueText;
1330 ControlOctetStrEdit.prototype.onTextInputChange = _ControlOctetStrEdit_OnTextInputChange;

1335 // *****
1340 // Private Methods prototypes
1345 // ***** /

```

1245 controlOctetStrEdit.prototype.formatValue = _controlOctetStrEdit_FormatValue;

1250 /***** END ControlOctetStrEdit Class *****/

Exhibit 11

```

1  /*=====
    AUTOMATED LOGIC CORPORATION
    Copyright (c) 1999 - 2000 All Rights Reserved
    This document contains confidential/proprietary information.
5  =====

    @(#)PrimitiveUpdateServlet.java

10  Author(s) ddavidson
    $Log: /green/source/com/controlj/green/web/servlets/PrimitiveUpdateServlet.java $
    * 10 12/09/99 1:50p Jomisoore
    * Made getChangesFromCore a method in ServletBase.
15  * 9 11/24/99 1:32p Jomisoore
    * Added more reserved parameters for callbacks.
    * 8 11/22/99 2:18p Tirish
20  * 7 10/28/99 4:16p Ddavidson
    * added 'false' argument to req.getSession( false ) to avoid created new
    * session when not wanted.
    * 6 10/11/99 11:53a Wdover
25  * 5 9/28/99 2:52p Wdover
    * 4 9/03/99 2:12p Wdover
30  * Modified to gracefully exit after HTTPSession timeout
    * 3 9/03/99 11:28a Sappling
    * Added Javascript call to trigger reload
    * 2 8/26/99 10:46a Ddavidson
35  * Added Submit/Update servlet for primitives.
    * Moved stdHeader/Footer/Content methods into ServletBase

=====*/

40 package com.controlj.green.web.servlets;

    import java.io.*;
    import java.util.*;
45  import javax.servlet.*;
    import javax.servlet.http.*;
    import com.controlj.green.web.servlets.*;
    import com.controlj.green.web.servlets.util.*;
50  import com.controlj.green.web.coreaccess.PrimitiveBinder;

```



```

55  /**<!--=====
    PrimitiveUpdateServlet is responsible for retrieving action page changes
    from a PrimitiveBinder and generating the JavaScript code that will
    notify that client of the new values via JscriptSetMasterValue objects.
    =====>*/

    public class PrimitiveUpdateServlet extends ServletBase
    {
        60  /**<!--===== doGet =====>
            Handle HTTP GET requests.  Serves resource after verifying session.
            The resources is any file specified by the "path" following the
            servlets URL.  For example "/servlet/protector/dir/x.htm" will serve
            the "dir/x.htm" file in the configured PRO_PROTDIR location.

            65  @param req  The standard HttpServletRequest
                @param resp The standard HttpServletResponse
                @author  ddavidson

                <!--=====*>

            70  public void doGet(HttpServletRequest req, HttpServletResponse resp)
                throws ServletException, IOException
                {
                    String afterUpdateCallback = null;

                    75  debugInfo( "PrimitiveUpdate:doGet - HTML" );
                        //////////////////////////////////////////
                        ////////////////////////////////////////// Extract parameters
                        //////////////////////////////////////////
                        String getValuesFromField = req.getParameter( "getFieldValues" ) ;

                        80  webBrowserSession wbs = null;
                            try
                            {
                                wbs = verifySession( req, true );
                                85  }
                                catch (ServletException e)
                                {
                                    if (req.getSession(false) == null)
                                    {
                                        90  redirectTimeout(resp);
                                            return;
                                    }
                                    else
                                    {
                                        95  throw e;
                                    }
                                }

                                afterUpdateCallback = req.getParameter("cjAfterUpdateCallback");

                                PrimitiveBinder pb = (PrimitiveBinder)wbs.get( "PrimitiveBinder" );
                                100

```

```

105  JscriptMsgList list = new JscriptMsgList( );
    // we are responsible for PrimitiveBinder synchronicity
    if ( pb != null )
    {
        getChangesFromCore(pb, getValuesFromField, list);
    }

110  list.addItem( new JscriptSimple("top.setActBusy(false)"););

115  if (afterUpdateCallback != null)
    {
        list.addItem( new JscriptSimple(afterUpdateCallback + "();"));
    }
    else
    {
120  list.addItem( new JscriptSimple("window.parent.updateFinished();"));
    }

    //////////////////////////////////////////////////
    // Now generate output stream and setup content
    //////////////////////////////////////////////////

125  stdContent( req, resp, null, list, null );
    }

130  /**<!===== doPost =====>
    POST redirects to "GET"
    @param resp The HttpServletResponse
    @author ddavidson

135  <!=====>*/

    public void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException
    {
        doGet( req, resp );
    }

    //////////////////////////////////////////////////
    // Standard debug
    //////////////////////////////////////////////////

145  private void debugInfo(String message) { com.controlj.green.common.Debug.info(
        "PrimitiveUpdateservlet",message); };
    private void debugError(String message) {
        com.controlj.green.common.Debug.error("PrimitiveUpdateservlet",message); };
    }

```

Exhibit 12

```

1 /*=====
   AUTOMATED LOGIC CORPORATION
   Copyright (c) 1999 - 2000 All Rights Reserved
   This document contains confidential/proprietary information.
5 =====

@(#)ServletBase.java

10 Author(s) ddavidson

   $Log: /green/source/com/controlj/green/web/servlets/ServletBase.java $
   * 16 12/09/99 1:48p Jomisoore
   * Added getChangesFromCore so that the submit and update servlets can
15 * both call it.
   *
   * 15 10/28/99 4:15p Ddavidson
   * fixed expireBrowserSession to handle 'null' login context
20 * 14 10/11/99 11:53a wdover
   *
   * 13 9/28/99 2:52p wdover
   *
   * 12 9/17/99 4:51p wdover
25 * Fixed problem with CJException
   *
   * 11 9/13/99 8:53a Ddavidson
   * changed login exception handling. removed relative path name in
   * redirectTimeout.
30 *
   * 10 9/10/99 10:22a Ddavidson
   * added getWebLoginContext() method
   *
   * 9 9/09/99 11:30a Ddavidson
35 * First pass at adding security via Operator and PrivilegeSet.
   * Nothing is checking privileges yet.
   *
   * 8 9/03/99 2:12p wdover
   * Adding redirectTimeout method
40 *
   * 7 8/26/99 5:02p Sappling
   *
   * 6 8/26/99 10:46a Ddavidson
   * Added Submit/Update servlet for primitives.
45 * Moved StdHeader/Footer/Content methods into ServletBase

```

```

* 5      8/24/99 9:41a Ddavidson
* Added "addStdJavaScriptReturn" method.
*
50 * 4      8/20/99 4:35p Ddavidson
* Removed local debug viewer
*
* 3      8/12/99 10:15a Ddavidson
* Rewrite using new core for Milestone 2
55 *
* 2      8/09/99 2:46p Ddavidson
* web browser session
*
* 1      6/30/99 9:27a Ddavidson
60 =====*/

package com.controlj.green.web.servlets;

65 import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.controlj.green.common.*;
70 import com.controlj.green.web.coreaccess.*;
import com.controlj.green.web.servlets.util.*;
import com.controlj.green.core.data.*;

/**<!--=====
75 Base class under which all swweb specific servlets should derive
themselves. Contains helper methods that are available to all servlet
subclasses. Before serving up a session specific data, the servlet
should call the verifySession() method to verify that the current
session, if any, is valid. If so, a WebBrowserSession will be returned
that may be used to manage any user-session specific information.
80 <P>
For each HttpSession, there is exactly one WebLoginContext object.
There may be many WebBrowserSession objects as well. There should be exactly
one for each browser window that is open at a time. Each servlet
85 request must identify itself with a "wbs=#" parameter as part of each
request. If a browser window is "duplicated", a new user context ID
will be assigned to the new window as part of the JavaScript response.
<P>
90 VerifyContext will automatically handle the case where a user performs
a "browser: open new window". A second window requires a new user
context.
<P>

```



```

140  @param req      The request triggering the login
    @param sc      The login servlet's configuration. Provides session global params.
    @param user    The user name to verify.
    @param pass    The supplied user password.
    @return a WebLoginContext if login was successful, else 'null'.
    @author ddavidson
145  <!--=====*>

    protected WebLoginContext login( HttpServletResponse req, ServletConfig sc,
        throws com.controlj.green.common.CJException
150  {
        // be sure to clean up any old info
        // invalidateSession( req );

        Operator operator = Operator.getOperator( user, pass );
        user = operator.getLoginName();
155
        // create a new session object and a WebLoginContext
        HttpSession session = req.getSession( true );
        return createWebLoginContext( req, sc, session, operator, user );
160  }

    /**<!--===== verifySession =====>
        Verify that we have a valid session and return the WebBrowserSession.
165
        Verifies that an HttpSession exists and that it contains a
        WebBrowserSession. Throws an exception if either is invalid. The request
        must contain a parameter identifying the WebBrowserSession via "wbs".

        @param req  The HttpServletResponse to be verified
        @param create True if we should create a new WebBrowserSession if one not found.
        @return a WebBrowserSession
        @author ddavidson
170  <!--=====*>

    static protected WebBrowserSession verifySession( HttpServletResponse req,
        boolean create )
        throws ServletException
175  {
        return WbsBean.verifySession( req, create );
180  }

    /**<!--===== invalidateSession =====>
        Invalidates the current HttpSession.

```

```

185 Further access using the current session ID will fail. This should
    not normally be called by a servlet.

190 @param req
    @author ddavidson
    <!--=====*>/

protected void invalidateSession( HttpServletRequest req )
{
    HttpSession session = req.getSession( false );
    if ( session != null )
        session.invalidate();
}

200 /**<!--===== expireWebBrowserSessions =====>
    Scan the WebBrowserSession for this HttpSession and destroy any that are
    older than out timeout property.
    @param req      HttpRequest triggering the scan
    @return a boolean true if there are any WebBrowserSession still active
    @author ddavidson
    <!--=====*>/

205 protected boolean expireWebBrowserSessions( HttpServletRequest req )
{
    HttpSession session = req.getSession( false );

    int numLeft = 0;
    webLoginContext lc = null;
    if ( session != null )
    {
        lc = (WebLoginContext)session.getValue( WebBrowserSession.WBS_KEY_LOGINCONTEXT );

        if ( lc != null )
        {
            String timeoutStr = (String)lc.get( lc.PROP_TIMEOUT );
            int to = 300000; // default to 5 minutes (in milliseconds)
            if ( timeoutStr != null )
                to = Integer.parseInt(timeoutStr) * 1000;

            // search over all session properties and look for any that
            // are our WebBrowserSession.
            String names[] = session.getValueNames();
            for ( int i=0; i<names.length; i++ )
            {
                if ( names[i].startsWith( WebBrowserSession.WBS_KEY_BROWSERSESSION ) )

```



```

235 {
    webBrowserSession wbs = (webBrowserSession)session.getValue( names[i] );
    if ( wbs.getLastAccessInterval() >= to )
    {
        session.removeValue( names[i] );
    }
    else
    {
        numLeft++;
    }
}
240 }
}
return (numLeft > 0);
245 }

/**<===== getWebLoginContext =====>
Find the web login context, if one exists for the current HTTP session.
@param req The HttpServletRequest to search.
@return a webLoginContext for the given session, or null if not found.
@author ddaavidson
<=====*>*/
protected webLoginContext getWebLoginContext( HttpServletRequest req )
{
    // get session, but don't create if not found
    HttpSession session = req.getSession( false );
    if ( session != null )
    {
        return (webLoginContext)session.getValue(
260             webBrowserSession.WBS_KEY_LOGINCONTEXT );
    }
    return null;
}

265 /**<===== createWebLoginContext =====>
Create a webLoginContext for the current session. This webLoginContext
will be available to all servlets for this session. It is placed
under a "well known" key (ie. "com.controlj") in the HttpSession.

@param req The request triggering the login
@param sc The servlet's configuration (this should be the login servlet)
270 @param session The HttpSession under which data is stored.
@param security The core's CorbaLoginContext to login under
@param user User name doing the login.
@return a webLoginContext
275

```

```

280      @author ddavidson
      <!--=====*>

      private webLoginContext createWebLoginContext(
      HttpServletRequest req, ServletConfig sc, HttpSession session,
      Operator operator, String user )
      {
          285      // construct the actual webLoginContext
          webLoginContext lc = new webLoginContext( operator,
          session.getId(), user + "@" + req.getRemoteAddr() );

          // place it into the HttpSession object
          290      session.putValue( webBrowserSession.WBS_KEY_LOGINCONTEXT, lc );

          // extract properties from the servlet config and place into user context
          Enumeration enum = sc.getInitParameterNames();
          while ( enum.hasMoreElements() )
          {
              295      // enumerate over all servlet's config props and put in the ones we know
              boolean found = false;
              String name = (String)enum.nextElement();
              String p = sc.getInitParameter( name );
              for ( int i=0; i<webLoginContext.propList.length; i++ )
              {
                  300      if ( name.equalsIgnoreCase( webLoginContext.propList[i] ) )
                  {
                      lc.put( webLoginContext.propList[i], p );
                      found = true;
                  }
              }
              if ( ! found )
              {
                  305      log( "Property name [" + name + "] is unexpected" );
                  // FIX XXX do we care about the ones we don't recognize?
              }
          }

          310      // set the session specific timeout value if specified in the config
          String p = (String)lc.get( lc.PROP_TIMEOUT );
          if ( p != null )
          {
              session.setMaxInactiveInterval( Integer.parseInt(p) );
          }

          315      return lc;
      }

      320      /**<!--===== addStdJavaScriptReturn =====>

```

```

325 Insert anything in the "jsreturn" request parameter as literal javascript
    code to the end of the current list.
    @param req The HttpServletRequest object.
    @param list The jsriptMsgList response list.
    @author ddavidson
    <!--=====*>/

330 protected void addStdJavaScriptReturn( HttpServletRequest req, jsriptMsgList list )
    {
        String js = req.getParameter( "jsreturn" );
        if ( js != null )
        {
            // add item as literal text to the end
            list.addItem( new jsriptSimple( js ) );
        }
    }

335

340 /**<!--===== stdHeader =====>
    Output our standard TreeRequest HTML header for JavaScript response.
    @param out HTML output stream
    @param wbs The current webBrowserSession
    @param seq The JavaScript supplied sequence number
    @author ddavidson
    <!--=====*>/

345 protected void stdHeader( PrintWriter out, WebBrowserSession wbs, String seq )
    {
        out.println("<HTML>");
        out.println("<SCRIPT LANGUAGE=\"JavaScript\">");
        out.println("// Time served is " + new Date() );

        // if the user context is new, tell the java script what it's ID is
        if ( wbs != null && wbs.isNew() )
            out.println("window.parent.setWBS( " + wbs.getId() + " )");

        if ( seq != null )
        {
            out.println("if (window.parent.reqStarted( " + seq + " )){");
        }
    }

350

355

360

365 /**<!--===== stdFooter =====>
    Output our standard HTML footer for JavaScript response.
    @param out HTML output stream
    @param seq The JavaScript supplied sequence number
    @param ack Our acknowledgement sequence number.

```

```

370 @author ddavidson
    <!--=====*>/
protected void stdFooter( PrintWriter out, String seq, String ack )
{
    if ( seq != null && ack != null) // tree request
    {
        out.println("window.parent.reqFinished( " + seq + ", " + ack + " )\n}");
        out.println("top.setNavBusy(false)");
    }
    out.println("</SCRIPT></HTML>");
}
380 }

/**<!--==== stdContent =====>
Output standard page response. Sets HTML content and cache control,
calls stdHeader(), writes the comment, outputs the list, calls stdFooter()
and then closes the output stream.
385

@param req The HttpServletRequest
@param resp The HttpServletResponse
@param wbs The current WebBrowserSession
390 @param list The JScriptMsgList of items to send
@param comment And arbitrary javascript comment to append
@author ddavidson
<!--=====*>/

protected void stdContent( HttpServletRequest req, HttpServletResponse resp,
    WebBrowserSession wbs, JScriptMsgList list, String comment)
    throws IOException
{
    resp.setContentType("text/html");
    resp.setHeader( "Cache-Control", "no-cache" );
    PrintWriter out = new PrintWriter(resp.getOutputStream());
    String seq = req.getParameter( "seq" );
    // insert any standard javascript footer code
    addStdJavaScriptReturn( req, list );
    // write standard header
    stdHeader( out, wbs, seq );
    // include any comment
    if ( comment != null )
        out.println( comment );
410 }

```

```

415 // write the body
    if (list != null)
        list.write( out );
    // write standard footer
420 stdFooter( out, seq, (list != null) ? list.getSequence() : "" );
    // and flush the output
    out.close();
425 } /**<===== redirectTimeout =====>
    gracefully handles HTTPSession timeouts.
    @param resp HttpServletResponse
    @param seq The JavaScript supplied sequence number
    @param ack Our acknowledgement sequence number.
    @author ddavidson
430 <!--=====
protected void redirectTimeout (HttpServletResponse resp)
    throws IOException
    {
435     resp.setContentType("text/html");
        resp.setHeader( "Cache-Control", "no-cache" );
        PrintWriter out = new PrintWriter(resp.getOutputStream());

        out.println("<HTML>");
440        out.println("<SCRIPT LANGUAGE=\"JavaScript\">");
        out.println("window.top.location.href=\"/common/navpane/timeout.html\"");
        out.println("</SCRIPT></HTML>");
        out.close();
445    }

    /**<===== getChangesFromCore =====>
        Gets the changes from the primitive binder and sends them down to
        javascript.
450    @param PrimitiveBinder pb          - Primitive Binder (what else did you think).
        @param String getValuesFromField - Whether to get the values from the
            field ("true" or "false").
        @param JscriptMsgList list      - List to add the jscript code to.
455    @author jomisore

    <!--=====
protected void getChangesFromCore(PrimitiveBinder pb,
    String getValuesFromField,
460

```

```

                                JscriptMsgList list)
{
    synchronized ( pb )
    {
        vector changes;
        if ((getValuesFromField != null) && getValuesFromField.equals("true"))
        {
            changes = pb.getPrimitiveValueChangesFromField();
        }
        else
        {
            changes = pb.getPrimitiveValueChanges();
        }
        Enumeration enum = changes.elements();
        while ( enum.hasMoreElements() )
        {
            PrimitiveBinder.PrimitiveValueChange change =
                (PrimitiveBinder.PrimitiveValueChange)enum.nextElement();
            list.addItem( new JscriptSetMasterValue( change.id, change.value ) );
        }
    }

    ////////////////////////////////////// Standard debug //////////////////////////////////////
    static private void debugInfo(String message) { com.controlj.green.common.Debug.info(
        "servletBase",message); };
    static private void debugError(String message) {
        com.controlj.green.common.Debug.error("servletBase",message); };
}

```

Exhibit 13

```

1  /*=====
    AUTOMATED LOGIC CORPORATION
    Copyright (c) 1999 - 2000 All Rights Reserved
    This document contains confidential/proprietary information.
5  =====

    @(#)JscriptMsgList.java
    Author(s) ddavidson
10   $Log: /green/SOURCE/COM/controlj/green/web/servlets/util/JscriptMsgList.java $
    * 2      8/12/99 10:15a Ddavidson
    * Rewrite using new core for Milestone 2
15   * 1      6/30/99 9:27a Ddavidson
    =====*/

20 package com.controlj.green.web.servlets.util;

    import java.util.*;
    import java.io.PrintWriter;

25 /**<!--=====
    Contains a list of JscriptMsg objects and manages insertion order
    and output of the entire list. Create items, add them, and write them.
    @author ddavidson.
    <!--=====*>*/

30 public class JscriptMsgList
    {
        /**<!-- seq ----->
        Current request sequence id we are sending items for.
        <!--=====*>*/
        private String seq;

        /**<!-- items ----->
        List of JscriptMsgs.
        <!--=====*>*/
        private Vector items;

        /**<!--===== JscriptMsgList =====>
        Create empty list with the associated sequence value.
        @param seq
        @author ddavidson
        <!--=====*>*/

50     public JscriptMsgList( String seq )
    {

```



```

    this.items = new Vector();
    this.seq = seq;
}

55  /**<===== JscriptMsgList =====>
    Create empty list with an empty sequence value for items that don't
    request future acknowledgement.
    @author ddavidson
    <!--=====*>*/

60  public JscriptMsgList()
    {
        this( "" );
    }

65  /**<===== getSequence =====>
    Return the request sequence value this list is associated with.
    @return a String
    @author ddavidson
    <!--=====*>*/

70  public String getSequence()
    {
        return seq;
    }

75

80  /**<===== addItem =====>
    Adds item to list based on requested order by the item.
    Set's the items sequence value to be that of this containing list.
    @param item
    @author ddavidson
    <!--=====*>*/

85  public void addItem( JscriptMsg item )
    {
        // if item has an insertion order the determine where to insert it
        if ( item.hasInsertOrder() )
        {
            for ( int i=0; i<items.size(); i++ )
            {
                if ( item.insertBefore( items.elementAt(i).getClass() ) )
                {
                    items.insertElementAt( item, i );
                    item = null; // don't fall thru to tail insertion
                    break;
                }
            }
        }
        // If still around, then add it to the end.
100

```

```

105     if ( item != null )
106     {
107         items.addElement( item );
108     }
109 }
110
111 /**<!===== write =====>
112     Invokes "encodeHtml( PrintWriter )" on each item in order.
113     @param out The PrintWriter to output the items to.
114     @author ddavidson
115     <!======*>*/
116
117 public void write( PrintWriter out )
118 {
119     Enumeration enum = items.elements();
120     while ( enum.hasMoreElements() )
121     {
122         jsriptmsg j = (Jsriptmsg)enum.nextElement();
123         j.encodeHtml( out );
124     }
125 }

```

Exhibit 14

[illegible]

[illegible]

```

95 public void encodeHtml( PrintWriter out )
   {
   }
   out.println( jscript.toString() );

100 /**<===== getText =====>
    Return the current text.
    @return a String
    @author ddavidson
    <!=====*>

105 public String getText()
   {
   }
   return jscript.toString();

110 /**<===== hasInsertOrder =====>
    Default doesn't care - add to end of list. Override in subclass.
    @return false
    @author ddavidson
    <!=====*>

115 public boolean hasInsertOrder()
   {
   }
   return false;

120 /**<===== insertBefore =====>
    Doesn't care - always return false. Override in subclass.
    @param testClass The test class
    @return a boolean Always false.
    @author ddavidson
    <!=====*>

125 public boolean insertBefore( Class testClass )
   {
   }
   return false;

130 /**<===== add =====>
    Add the literal string to the text with no formatting.
    @param s The string to add.
    @author ddavidson
    <!=====*>

```

```

140 public void add( String s )
141 {
142     jscript.append(s);
143 }

145 /**<!==== addQuotedArg =====>
    Add the string as a quote argument, handling commas where necessary.
    @param s The string to add.
    @author ddavidson
    <!=====*>*/

150 public void addQuotedArg( String s )
151 {
152     if (isFirstArg )
153     {
154         isFirstArg = false;
155     }
156     else
157     {
158         jscript.append( ", " );
159     }
160     jscript.append( "'" );
161     jscript.append( CJIO.js(s) );
162     jscript.append( "'" );
163 }

165 /**<!==== addQuotedArg =====>
    Add the int as a quote argument, handling commas where necessary.
    @param i The integer to add.
    @author ddavidson
    <!=====*>*/

170 public void addQuotedArg( int i )
171 {
172     addQuotedArg( Integer.toString(i) );
173 }

175 /**<!==== addQuotedArg =====>
    Add the long as a quote argument, handling commas where necessary.
    @param l The long to add.
    @author ddavidson
    <!=====*>*/

180 public void addQuotedArg( long l )
181 {
182     addQuotedArg( Long.toString(l) );
183 }

```

```

    }

    /**<!= addArg =====>
    Add the string as an unquoted argument, handling commas where necessary.
    @param s The String to add.
    @author ddavidson
    <!======*/

    public void addArg( String s )
    {
        // add a comma before all but the first arg on a line
        if (isFirstArg )
        {
            isFirstArg = false;
        }
        else
        {
            jscript.append( ", " );
        }
        jscript.append( s );
    }

    /**<!= addArg =====>
    Add the int as an argument, handling commas where necessary.
    @param i The integer to add.
    @author ddavidson
    <!======*/

    public void addArg( int i )
    {
        addArg( Integer.toString(i) );
    }

    /**<!= addArg =====>
    Add the long as an argument, handling commas where necessary.
    @param l The long to add.
    @author ddavidson
    <!======*/

    public void addArg( Long l )
    {
        addArg( Long.toString(l) );
    }

    /**<!= addArg =====>
    Add the boolean as an argument, handling commas where necessary.

```



```

235      @param b The boolean to add.
      @author ddavidson
      <!--=====*>/
      public void addArg( boolean b )
      {
          addArg( b ? "true" : "false" );
      }
240
      /**<!--===== addNewline =====>
      Add a newline to the current text, handling commas where necessary.
      @author ddavidson
      <!--=====*>/
245      public void addNewline()
      {
          // If no argument on this line yet, just add the newline,
          // else append a command and start a new line
          if ( isFirstArg )
          {
              jscript.append( '\n' );
          }
          else
          {
              jscript.append( ",\n" );
              isFirstArg = true;
          }
255      }
260
      /**<!--===== addNewline =====>
      Same as addNewline() but inserts the prefix text on the next line.
      This is useful for indenting subsequent lines.
      @param nextPrefix
      @author ddavidson
      <!--=====*>/
265      public void addNewline( String nextPrefix )
      {
          addNewline();
          jscript.append( nextPrefix );
      }
270
      /**<!--===== startMethod =====>
      Starts a method declaration with the specified name.

```

Appends a "(" and after the specified name. The name may include any text, for example "foo = new func".


```
280 <b>WARNING: Does not support nested methods currently.</b>.
```

```
285  @param name
    @author ddavidson
    <!--=====*>

    public void startMethod( String name )
    {
        jscript.append( name );
        jscript.append( "( " );
        isFirstArg = true;
    }

    /**<!--===== endMethod =====>
    Ends a method declaration, optional adding a semicolon after ").
    If ending a nested method, specify false.
    @param addsemicolon True if a semicolon should be added after the
        function close.
    @author ddavidson
    <!--=====*>

    public void endMethod( boolean addsemicolon )
    {
        if ( addsemicolon )
        {
            jscript.append( " );\n" );
        }
        else
        {
            jscript.append( " )" );
        }
        isFirstArg = true;
    }

    /**<!--===== beginList =====>
    Mark the current state at the beginning of a comma separated list.
    @return a String
    @author ddavidson
    <!--=====*>

    public void beginList()
    {
        isFirstArg = true;
    }
}
```

```
325     }
    /**<!= ===== toString =====>
        Return the HTML output as a string.
        @return a String
        @author ddavidson
    330 <!= =====> */
    public String toString( )
    {
    335     return javascript.toString();
    }
}
```

Exhibit 15

```

1 /*=====
   AUTOMATED LOGIC CORPORATION
   Copyright (c) 1999 - 2000 All Rights Reserved
   This document contains confidential/proprietary information.
5 =====

   @(#)JscriptSetMasterValue.java

10   Author(s) ddavidson

   $Log: /green/SOURCE/COM/controlj/green/web/servlets/util/JscriptSetMasterValue.java $
   * 2 8/26/99 10:46a Ddavidson
   * Added Submit/Update servlet for primitives.
15 * Moved stdHeader/Footer/Content methods into ServletBase
   *
   * 1 8/26/99 8:26a Ddavidson
   =====*/

20   package com.controlj.green.web.servlets.util;

   import java.io.*;

25   /**<!--
   A JscriptSimple derived class to send primitive value change messages.
   They insert at the tail of a list.
   @author ddavidson.
30   <!--=====*>*/

   public class JscriptSetMasterValue extends JscriptSimple
   {

35   /**<!--===== JscriptSetMasterValue =====>
   Inserts a PrimitiveBase.setPrimitiveMasterValue for the given primitive
   ID and value.
   @param id The primitive's ID.
   @param value The primitive's new value.
   @author ddavidson
40   <!--=====*>*/

   public JscriptSetMasterValue( String id, String value )
   {
45       super( "" );
       startMethod( "window.parent.PrimitiveBase.setPrimitiveMasterValue" );
       addQuotedArg( id );
       addQuotedArg( value );
       endMethod( true );
50   }

```